VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Segmentace obrazu s využitím metody grafových řezů

Image Segmentation by Making Use of the Graph Cuts Method

2012 Michael Hrabálek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Michael Hrabálek**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Segmentace obrazu s využitím metody grafových řezů
Image Segmentation by Making Use of the Graph Cuts Method

Zásady pro vypracování:

Segmentace obrazu je významným krokem při jeho zpracování. Cílem diplomové práce je prakticky ověřit vlastnosti metody popsané v článku Gorelick L. et al.: "Recursive MDL via grath cuts: application to segmentation" (ICCV 2011). V diplomové práci proveďte:

1. Prostudujte výše uvedený článek a články související.
2. Popisovanou metodu naimplementujte v jazyce C/C++.
3. Prověřte vlastnosti metody na serii experimentů.
4. Podle okolností se můžete pokusit metodu i vylepšit.
5. Dosažené výsledky pečlivě zdokumentujte v textové části práce.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Dr. Ing. Eduard Sojka**

Datum zadání: 18.11.2011

Datum odevzdání: 04.05.2012

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 4. května 2012

.....

Děkuji všem, kteří mi s touto prací pomohli, zejména mému vedoucímu práce panu doc. Dr. Ing. Eduardu Sojkovi za jeho čas, odborné vedení a množství rad, které mi ochotně poskytl.

Abstrakt

Tato práce se zabývá a ověřuje novou metodu segmentace obrazu, kterou zveřejnil Gorelick L. et al. v článku "Recursive MDL via Graph Cuts: Application to Segmentation" (ICCV 2011). Metoda poskytuje poněkud neobvyklou reprezentaci obrazu pomocí tzv. záplat. Tato reprezentace je velmi užitečná, jelikož jsou s její pomocí v obraze detekovány oblasti, ve kterých se opakuje nějaký vzor (záplata). K nalezení záplatové reprezentace je použito principu hledání minimální délky popisu, pro jehož řešení jsou využívány grafové řezy. V práci je uveden teoretický základ jak zmíněné segmentační metody, tak grafových řezů, o které se metoda opírá. Kromě teoretického popisu byla v rámci práce metoda naimplementována a ověřena na sadě experimentů, které jsou v práci zdokumentovány.

Klíčová slova: segmentace obrazu, grafový řez, minimální délka popisu, hierarchické splynutí, rekurze, záplatový model, alpha expanze

Abstract

This thesis deals with a new image segmentation method, published by L. Gorelick et al. in the article "Recursive MDL via Graph Cuts: Application to Segmentation" (ICCV 2011). The method provides a novel patch-based image representation. This representation is very useful because it detects the regions with repetitive pattern in image. The principle of finding minimum description length, which is determined by making use of the graph cut algorithm, is applied to find the patch-based representation. The thesis presents the theoretical basis for both the segmentation method as well as for the graph cuts, which are the basis areas in this context. In addition to the theoretical description, the method is implemented and verified on a set of experiments that are documented in the thesis.

Keywords: image segmentation, graph cut, minimum description length, hierarchical fusion, recursion, patch-based model, alpha expansion

Seznam použitých zkratek a symbolů

GC	–	graph cuts
CT	–	computer tomograph
MDL	–	minimum descriptioin length
RDAG	–	rooted directed acyclic graph
HF	–	hierarchical fusion

Obsah

Seznam obrázků

1 Úvod

Významnou součástí digitálního zpracování obrazu je jeho segmentace. Pro segmentaci již bylo navrženo mnoho metod. Žádná z nich však není považována jako univerzální. Pro každou jsou vhodné určité typy obrázků. Já se v této práci zabývám a prakticky ověřuji novou segmentační metodu, kterou navrhli Gorelick L. et al. v článku "Recursive MDL via Graph Cuts: Application to Segmentation"(ICCV 2011). Tato metoda je vhodná pro segmentaci obrázků, ve kterých se vyskytují oblasti s nějakým opakujícím se vzorem. Pod takovým obrázkem si představte třeba fotografii polí s tulipány, kde mají tulipány v každém poli jinou barvu. Opakujícím se vzorem je v tomto případě různě barevný tulipán a segmentace by měla od sebe oddělit pole, ve kterých mají tulipány různé barvy.

Zmíněný opakující se vzor je jinak označován jako záplata. Výhodou je, že záplata není ničím jiným než obrázkem a lze na ní aplikovat stejnou myšlenku segmentace. Takovým rekurzivním procesem vzniká tzv. záplatový model obrazu. K nalezení tohoto modelu je použito principu hledání minimální délky popisu, k jehož řešení jsou použity grafové řezy.

Text práce začíná stručným úvodem do segmentace. Po něm následuje kapitola týkající se grafových řezů. Jelikož jsou grafové řezy zásadní pro popisovanou segmentační metodu, je jejich popis poměrně podrobný. V další kapitole se nachází detailní výklad segmentační metody, kterou se zabývám. Cílem práce bylo také tuto metodu naimplementovat a prakticky ověřit její funkčnost. V páté kapitole je proto zdokumentována sada experimentů, které jsem provedl pomocí vlastní implementace. V šesté kapitole uvádím stručný komentář k implementaci.

2 Segmentace obrazu

2.1 Úvod

V oblasti analýzy digitálního obrazu je segmentace jedním z prvních kroků. Jedná se o proces, kdy je vstupní obraz rozdělen do několika oblastí (segmentů). Každou oblast tvoří množina pixelů, které spolu nějakým způsobem souvisí. Například barvou, jasem, podobností textury. Jednotlivé oblasti se nesmí překrývat a dohromady pokrývají celý obraz. Díky segmentaci získáme jinou reprezentaci obrazu, která umožní snadnější a lepší zpracování v dalších krocích obrazové analýzy. Jak hned ukážu, výstupem segmentace nemusí být jenom oblast. Může se taky jednat o hranici objektu popsanou přímkami, křivkami atd..

Později v textu uslyšíte pojem značka. Té se využívá v takové definici segmentace, kdy je každému pixelu obrázku pomocí nějakého algoritmu přiřazena právě jedna značka a pixely se stejnou značkou mají podobné vizuální vlastnosti a tedy dohromady tvoří oblast (segment).

Pro segmentaci již bylo navrženo mnoho algoritmů. Dosud však nebyl nalezen žádný univerzální algoritmus, který by byl vhodný pro různé druhy obrazů a odlišné aplikace. Proto je nutné, pro každou aplikaci a nějakou skupinu obrazů, zvlášťcitlivě vybrat vhodný algoritmus.

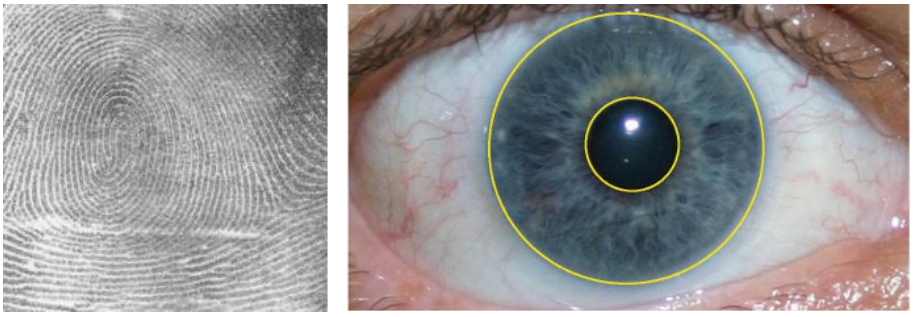
2.2 Využití

Analýza obrazu a její nedílná součást segmentace je využívána v mnoha odvětvích. Se zdokonalováním segmentačních metod, využitím většího výpočetního výkonu a dalšími inovacemi - roste i počet odvětví použití. Zmíním alespoň několik, které nás obklopují.

Biometrie

Biometrie je dnes už velice rozšířený způsob pro autentizaci. Využívá jedinečnosti biologických charakteristik člověka. Mezi autentizace, které využívají segmentace, patří

- detekce otisku prstu
- detekce obličeje
- detekce oční duhovky
- detekce oční sítnice

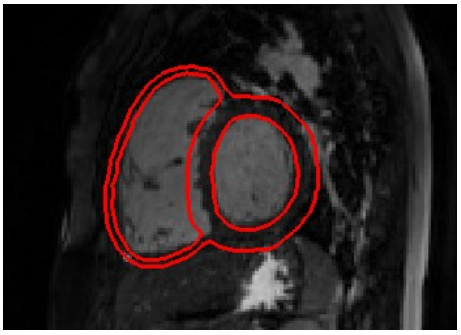


Obrázek 2.1: Vlevo je ukázka vstupního obrazu otisku prstu pro segmentaci. Na obrázku vpravo je výsledek segmentace (mezikruží) fotografie oka pro získání duhovky. Zdroj: [14, 15]

Zpracování lékařských snímků

V tomto odvětví jde většinou o to automaticky nalézt na snímcích něco, co tam normálně nepatří (např. nádor, zlomenina). Segmentují se jakékoliv medicínské snímky, at' už se jedná o snímky rentgenové, CT, magnetické rezonance, ultrazvuku, endoskopických kamer apod..

Další skvělou vlastností např. magnetické rezonance je, že poskytuje sadu snímků. Pokud se na každém z nich aplikuje segmentace a segmentované oblasti se nějakým dalším algoritmem sloučí, získáme 3D model.



Obrázek 2.2: Ukázka segmentace srdce na snímku magnetické rezonance, Zdroj: [13]

Doprava

V současnosti je v největším rozmachu automatické řízení auta. S tím souvisí mnoho obrazového zpracování. Např.

- detekce okrajů vozovky

- detekce dopravních značek
- detekce okolních vozidel

Další aplikace segmentace obecně v dopravě jsou např.

- počítání automobilů
- detekce SPZ (automatické rozmazání, automatické otevírání vrat)
- detekce chodců

To byl jen stručný výčet oblastí, kde se segmentace obrazu využívá. Některá využití už běžně kolem sebe vídáme, některá jsou v počátcích. Od této praktické ukázky se přesunu zpět k popisu segmentace. Zmínil jsem, že existuje mnoho technik pro segmentaci. Ve stručnosti je rozdělím do několika kategorií a každou krátce popíšu.

2.3 Základní dělení

Způsobů, jak rozdělit druhy segmentací, je více. Můžeme je dělit třeba na lokální či globální. Podle způsobu výpočtu například na sekvenční a paralelní. Jestli pracují automaticky či s nějakou interakcí. Já je rozdělím do 3 hlavních skupin založených na

- analýze pixelů
- hledání hran
- hledání oblastí

Analýza pixelů

Do této skupiny zařadím jednu z nejjednodušších, zároveň však velmi rychlou metodu, v české literatuře zvanou, prahovaní (v anglické literatuře známá jako thresholding).

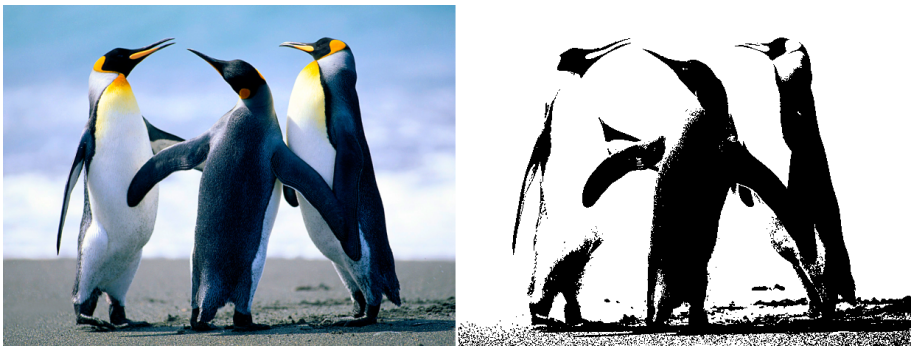
Prahování V samotném základu tato metoda prochází všemi pixely obrazu a porovnává jas v daném pixelu a stanovený práh. Je-li jas nižší než práh, je aktuální pixel označen jako pixel pozadí. Naopak pokud je práh vyšší, je pixel označen jako patřící objektu. Popřípadě se můžou pixely pozadí označit jako patřící objektu a naopak. Výsledkem je tedy obraz, který obsahuje pouze dvě hodnoty - je tedy binární.

Zatím jsem nezmínil jednu podstatnou věc. A to jak se stanoví práh. Na začátku kapitoly jsem uvedl, že podle aplikace volíme vhodnou segmentační metodu. Tato volba pokračuje i uvnitř metody. V tomto případě, pokud je nutné, aby metoda pracovala automaticky, tak je práh počítán např. jako průměr nebo medián jasů všech pixelů nebo

vypočten nějakou lepší metodou na základě znalosti histogramu. Někdy je vhodné využít lidského vnímání a práh určit manuálně.

Co když ale potřebujeme více než jen rozdělení obrazu do dvou segmentů? Ani to není pro tuto metodu překážkou. Jednoduše přidáme další prahy a jasy pixelů testujeme na intervaly vzniklé mezi prahy.

Je vhodné abych uvedl i nějakou slabinu této metody. Neprůstřelná opravdu není. Například při špatném osvětlení scény v obraze může metoda některé části objektu chybně označit jako pozadí a naopak.



Obrázek 2.3: Ukázka použití prahování s automaticky nastaveným prahem v grafickém programu GIMP

Hledání hran

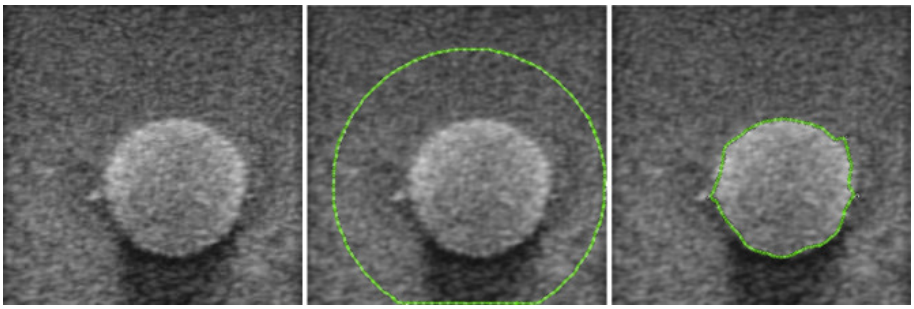
V této sekci uvedu dva zajímavé způsoby nalezení hrany objektu. Hranice se může skládat z několika úseček nebo třeba jen z jedné křivky. Prvním způsobem nalezení hrany, lépe řečeno sadou metod, jsou metody, které využívají první derivace jasové funkce.

Gradientní metody Tyto metody využívají skutečnosti, že v místě hrany má absolutní hodnota první derivace průběhu jasu vysokou hodnotu [17]. Velikost (intenzita) hrany je určena hodnotou derivace. Pro výpočet (aproximaci) gradientu bylo vyvinuto několik operátorů. Výhodou gradientních metod je rychlost. Slabinou jsou obrázky poškozené šumem, kdy vznikají hrany i tam, kde ve skutečnosti nejsou.

Druhým zajímavým způsobem nalezení hrany je využití tzv. aktivních kontur (active contours).

Aktivní kontury Pozadí této metody je poněkud obsáhlejší, nicméně metoda je velmi používaná, proto je vhodné ji alespoň zmínit. V principu je do obrázku položen had (množina spojených úseček) a ten pomocí minimalizace určité energie objektu objekt obejme. K výpočtu energie se mimo jiné využívá i gradientu.

Velkou výhodou této metody oproti gradientním metodám je, že funguje i na obrázcích poškozených šumem.



Obrázek 2.4: Praktická ukázka použití aktivní kontury. Na prvním obrázku je vidět původní obrázek poškozený šumem. Na druhém je inicializovaná aktivní kontura. Na třetím je vidět výsledná hrana.

Hledání oblastí

Oblasti jsou hledány díky určitému kriteriu homogenity. Např. podobná barva, jas, textura. Známými metodami jsou spojování a dělení oblastí.

Spojování a dělení oblastí U spojování (nebo také šíření) oblastí jsou na začátku malé oblasti. Můžou to být i samotné pixely. Jednotlivé oblasti se porovnávají a pokud splňují kriterium homogenity, tak jsou sloučeny do společné oblasti. Toto se opakuje, dokud je v obraze co spojovat.

U dělení je to obráceně. Začíná se s jednou oblastí, kterou tvoří celý obraz. Tato oblast se iterativně dělí, dokud není ve všech oblastech splněno kriterium homogenity. Zde je kriterium ale jiné. Používá se např. znalostí histogramu (obsahuje více vrcholů) nebo metoda diskriminantu [17].

3 Grafové řezy

3.1 Úvod

V předchozí kapitole jsem záměrně jednu skupinu segmentačních technik vynechal. Jedná se o využití grafových algoritmů, potažmo grafových řezů (graph cuts). Toto téma je zásadní pro algoritmus, kterým se v této práci zabývám, a proto bude vysvětleno poměrně podrobně. V následující části ukážu, jakým způsobem lze grafové řezy využít přímo k segmentaci obrazu. To však není jedinou možností grafových řezů. V podkapitole 3.3 ukážu použití grafových řezů k nalezení minima funkce (energie), a to pomocí algoritmu α-expanze. Závěrem se ještě podíváme, jak lze algoritmus α-expanze rozšířit.

3.2 Grafové řezy a segmentace

Aplikace grafových řezů umožňuje takovou segmentaci obrazu, která dělí obraz na dvě části - objekt a pozadí. Metoda, kterou popíšu, platí obecně pro N-rozměrný prostor. Může fungovat jak s interakcí, tak automaticky.

Během inicializace metody, at' už interakcí nebo automaticky, je nutné některé body označit jako body reprezentující objekt a některé reprezentující pozadí. Takto označené body značí tzv. tvrdá omezení a jsou použita dále ve výpočtu. Ještě existují tzv. měkká omezení založená na vlastnostech oblasti a hrany. Než se k nim dostanu, musím zavést potřebnou terminologii.

Nechť $\mathcal{P}$ označuje množinu všech pixelů obrazu a $\mathcal{N}$ značí množinu párů všech sousedících pixelů $\{p, q\}$. Dále nechť $\mathcal{O}$ značí množinu všech pixelů, které patří do objektu a $\mathcal{B}$ je množina všech pixelů patřících do pozadí. $A = (A_1, \ldots, A_p, \ldots, A_{|\mathcal{P}|})$ nechť je binární vektor, jehož komponenty A_p určují přiřazení pixelům $p \in \mathcal{P}$. Každé A_p může být "obj" nebo "bkg" (anglické zkratky pro objekt a pozadí). Samotný vektor A definuje výslednou segmentaci.

Měkká omezení založené na vlastnostech oblasti a hrany A jsou popsány cenovou funkcí $E(A)$ jako

$$E(A) = \lambda \cdot R(A) + B(A), \tag{1}$$

kde

$$R(A) = \sum_{p \in \mathcal{P}} R_p(A_p), \tag{2}$$

$$B(A) = \sum_{\{p,q\} \in \mathcal{N}} B_{p,q} \cdot \delta(A_p, A_q) \tag{3}$$

a

$$\delta(A_p, A_q) = \begin{cases} 1 & \text{pokud} A_p \neq A_q \\ 0 & \text{jinak.} \end{cases} \quad (4)$$

Koeficient $\lambda \geq 0$ určuje poměr vlastností oblasti $R(A)$ vůči vlastnostem hranice $B(A)$. Člen $R(A)$ předpokládá, že už jsou dány ceny odchylek vzniklé při zařazení pixelu p jako "obj" nebo "bkg", respektive členy $R_p("obj")$ a $R_p("bkg")$. Takže například $R_p(\cdot)$ může odrážet, jak odpovídá jas pixelu p histogramu objektu a pozadí.

Člen $B(A)$ zahrnuje vlastnosti hranice segmentace A. Koeficient $B_{\{p,q\}} \geq 0$ určuje cenu za nespojitost mezi p a q. Pokud jsou např. jasy v pixelech p a q podobné, je koeficient vysoký. Naopak pokud jsou jasy velice odlišné, blíží se koeficient nule. Pro výpočet může být použit např. některý z gradientních operátorů.

K minimalizaci funkce (1) a tedy k segmentaci jsou použity grafové řezy. Abych je mohl využít, musím nějakým způsobem převést obraz na graf. Vytvořím tedy ohodnocený graf $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, kde $\mathcal{V}$ je množina uzlů (vrcholů) a $\mathcal{E}$ množina neorientovaných hran, které spojují sousedící uzly. Jednotlivé uzly z množiny $\mathcal{V}$ odpovídají pixelům obrazu. Každé hraně $e \in \mathcal{E}$ v grafu je přiřazena určitá nezáporná cena w_e.

Kvůli tzv. tvrdým omezením (interakce uživatele) jsou ke grafu připojeny ještě další dva speciální uzly. Uzel "objektu"S a uzel "pozadí"T. Říká se jim také terminály. Množinu vrcholů můžeme tedy zapsat jako

$$\mathcal{V} = \mathcal{P} \cup \{S, T\}.$$

Terminály S a T jsou spojeny s uzly grafu, které reprezentují pixely obrazu tak, že ke každému uzlu existuje právě jedna hrana, která ho spojí s uzlem S a právě jedna hrana, která ho spojí s uzlem T. Množinu hran spojující mezi sebou uzly reprezentující pixely nazýváme $n - hrany$. Množinu hran, které spojují uzly reprezentující pixely s uzly S a T označujeme $t - hrany$. V tabulce č. 1 uvádím jednotlivé ceny hran.

Nyní známe všechno potřebné pro konstrukci grafu $\mathcal{G}$. Zbývá říct, co je řez grafu a jak s jeho pomocí minimalizuji funkci (1) a tedy provedu segmentaci.

3.2.1 Řez grafu

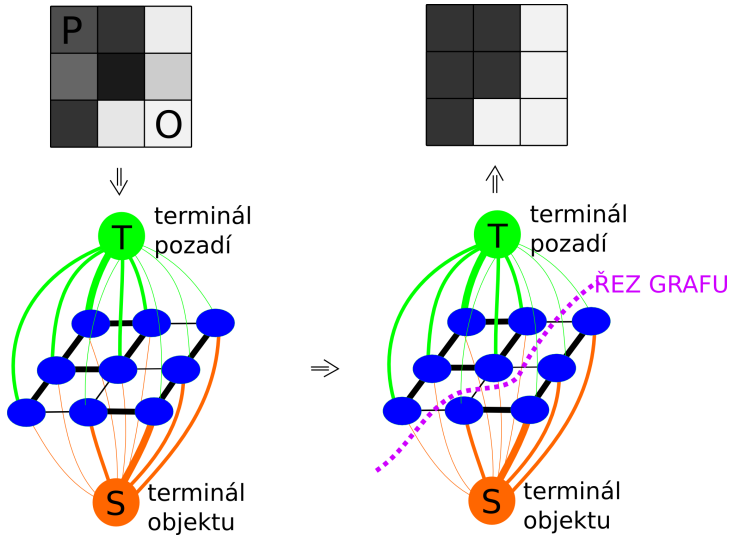
Řez $\mathcal{C}$ grafu $\mathcal{G}$ je taková podmnožina jeho hran $\mathcal{C} \subset \mathcal{E}$, která oddělí uzly S a T na grafu $\mathcal{G}(C) = \{\mathcal{V}, \mathcal{E} \backslash C\}$. Protože řez odděluje uzly S a T, nazývá se taky někdy s-t řezem. Cena řezu je definována jako součet cen hran, kterými prochází, což vede na vztah

$$|C| = \sum_{e \in C} w_e. \quad (5)$$

Grafovým řezem lze tedy obraz segmentovat. Zajímá nás ale, pro jakou cenu řezu dostaneme nejlepší výsledek segmentace. To vyslovím následující větou.

hrana	cena	pro
$\{p,q\}$	$B_{\{p,q\}}$	$\{p,q\} \in \mathcal{N}$
$\{p,S\}$	$\lambda \cdot R_p(\text{"bkg"})$	$p \in \mathcal{P}, p \notin \mathcal{O} \cup \mathcal{B}$
	K	$p \in \mathcal{O}$
	0	$p \in \mathcal{B}$
$\{p,T\}$	$\lambda \cdot R_p(\text{"obj"})$	$p \in \mathcal{P}, p \notin \mathcal{O} \cup \mathcal{B}$
	0	$p \in \mathcal{O}$
	K	$p \in \mathcal{B}$

Tabulka 1: Ceny hran grafu, kde $K = 1 + \max_{p \in \mathcal{P}} \sum_{q:\{p,q\} \in \mathcal{N}} B_{\{p,q\}}$.



Obrázek 3.1: Jednoduchá ukázka segmentace obrázku o rozměrech 3x3. Tloušťka hran znázorňuje jejich cenu. Na vstupním obrazu je písmenem P označen bod pozadí a písmenem O bod objektu.

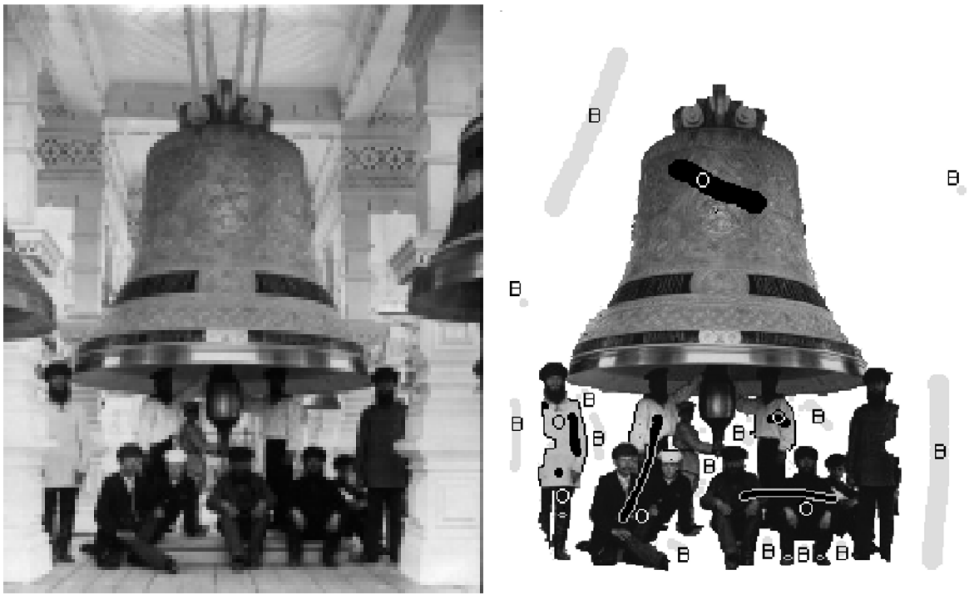
Věta 3.1 *S ohledem na možné řezy $\mathcal{F}$ na grafu $\mathcal{G}$, je nejvhodnější minimální řez $\hat{C}$ (řez s minimální cenou).*

Originální znění včetně důkazu je uvedeno v [18].

Nejlepší výsledek segmentace tedy dostanu minimálním řezem grafu $\mathcal{G}$. Poslední důležitou věcí už je jen znát způsob, jak minimální řez spočítat. Na řešení tohoto problému přišly v roce 1956 nezávisle na sobě dvě skupiny lidí. První tvořili L.R. Ford, Jr. a D.R. Fulkerson. Druhou P. Elias, A. Feinstein, and C.E. Shannon [20]. Všichni přišli na to, že je tento problém ekvivalentní problému maximálního toku v grafu (síti).

Věta 3.2 *Maximální hodnota toku s-t je ekvivalentní minimální řezu s-t s minimální cenou.*

Původní znění teorému včetně jeho důkazu je dostupné např. v [19].



Obrázek 3.2: Ukázka segmentace pomocí minimálního řezu grafu. Vlevo je originální obrázek. Vpravo je obrázek po segmentaci, kde uživatel předem vyznačil zárodky oblastí objektu O a pozadí B. Zdroj: [18]

3.2.2 Algoritmus maximálního toku

U problému nalezení maximálního toku z uzlu s do uzlu t se jedná o kombinatorickou optimalizační úlohu, kdy se hledá maximální průtok přes jednotlivé hrany grafu. Ceny jednotlivých hran určují maximální průtok (kapacitu) na dané hraně. Pro řešení tohoto problému bylo vyvinuto několik algoritmů. Já alespoň naznačím jeden z nich. Konkrétně algoritmus Ford-Fulkerson. Ten vymysleli v roce 1956 L.R. Ford, Jr. a D.R. Fulkerson.

Mějme graf $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ a pro každou hranu spojující uzly u a v kapacitu $c(u, v)$ a tok $f(u, v)$. Dále definujeme zbytkový graf $\mathcal{G}_f = \{\mathcal{V}, \mathcal{E}_f\}$ s kapacitami $c_f(u, v) = c(u, v) - f(u, v)$ a žádným tokem. Algoritmus začínáme s nulovým tokem. Tedy pro každou hranu platí $f(u, v) \leftarrow 0$. Potom se v iteraci hledá nejkratší cesta z s do t neprošlými hranami zbytkového grafu $\mathcal{G}_f$. Podél této cesty zvyšujeme tok, dokud není některá z hran nasycená. O kolik se zvýší tok v grafu $\mathcal{G}$, o tolik se sníží ve zbytkovém grafu $\mathcal{G}_f$. V každé iteraci se zvyšuje celkový tok z s do t, a to do té doby, než neexistuje žádná cesta s nenasycenými hranami. Tím dosáhneme maximálního toku. Původní minimální řez grafu tedy odpovídá nasyceným hranám grafu $\mathcal{G}$.

3.3 α-expanze

V předchozí kapitole 3.2 jsem hovořil o použití grafových řezů k segmentaci obrazu. Popsaný algoritmus byl schopen využít dva typy oblastí - objekt a pozadí. Jinými slovy grafové řezy hledaly optimální řešení binárního problému. Na začátku kapitoly 2 jsem zase zmínil, že se budeme bavit o nějakých značkách. Pojďme se tedy na ně podívat blíže.

Cílem je nalézt takové značení (zobrazení) f, které každému pixelu p z množiny všech obrazových pixelů $\mathcal{P}$ ($p \in \mathcal{P}$) přiřadí nějakou značku f_p z množiny všech značek $\mathcal{L}$ ($f_p \in \mathcal{L}$). Přičemž f hledáme takové, aby splnilo podmínky různých úkolů při zpracování obrazu (např. rekonstrukce, segmentace). Použití grafových řezů v předchozí kapitole 3.2 se dá také vysvětlit jako problém hledání značení f. Množinu $\mathcal{L}$ tvořily prvky "obj" a "bkg" a hledali jsme takové značení f, které ve výsledku dává vhodnou segmentaci obrazu.

3.3.1 Energie pro minimalizaci

Použití grafových řezů na více značkách je však výpočetně drahé [4]. Tento problém se snaží vyřešit algoritmus α-expanze za použití série grafových řezů. Algoritmus α-expanze hledá takové značení f, které minimalizuje energii ve tvaru

$$E(f) = E_{\text{data}}(f) + E_{\text{smooth}}(f), \tag{6}$$

kde $E_{\text{data}}(f)$ měří neshodu mezi značením f a pozorovanými daty a $E_{\text{smooth}}(f)$ je mírou rozsahu částí, kde je f nespojité. $E_{\text{data}}(f)$ je formulován jako

$$E_{\text{data}}(f) = \sum_{p \in \mathcal{P}} D_p(f_p), \tag{7}$$

kde funkce D_p měří, jak odpovídá značka pixelu p na pozorovaných datech. Například u problému rekonstrukce obrazu je

$$D_p(f_p) = (f_p - i_p)^2,$$

kde i_p značí jas pixelu p. Algoritmus α-expanze umožňuje ale libovolnou definici funkce D_p.

Funkce $E_{\text{smooth}}(f)$ je definována jako

$$E_{\text{smooth}} = \sum_{\{p,q\} \in \mathcal{N}} V_{\{p,q\}}(f_p, f_q), \tag{8}$$

kde $\mathcal{N}$ značí množinu párů všech sousedících pixelů. Na rozdíl od funkce D_p, nemůže být funkce $V_{\{p,q\}}$ libovolná, ale musí splňovat následující podmínky metriky

$$V(\alpha, \beta) = 0 \Leftrightarrow \alpha = \beta, \tag{9}$$

$$V(\alpha, \beta) = V(\beta, \alpha) \geq 0, \tag{10}$$

$$V(\alpha, \beta) \leq V(\alpha, \gamma) + V(\gamma, \beta). \tag{11}$$

První podmínka nám říká, že pokud mají značky α a β stejnou hodnotu, je výsledná energie nulová. Druhá podmínka značí, že energie musí být symetrická a nezáporná. Nakonec musí V splňovat ještě trojúhelníkovou nerovnost. Jedna z možných použitelných funkcí, která splňuje podmínky metriky je např. $V(\alpha, \beta) = min(K, ||\alpha - \beta||)$. Tedy minimum mezi nějakou pevnou reálnou konstantou K a funkcí euklidovské vzdálenosti.

3.3.2 Algoritmus a jeho vlastnosti

Nastíníme si algoritmus v krátkém pseudokódu a řekneme si některé základní vlastnosti. Jakékoliv značení f může být reprezentováno dělením obrazových pixelů $\mathbf{P} = \{\mathcal{P}_l | l \in \mathcal{L}\}$, kde $\mathcal{P}_l = \{p \in \mathcal{P} | f_p = l\}$ je podmnožina pixelů, které jsou označeny značkou l. Mezi značením f a oblastmi $\mathbf{P}$ je zřejmá totožnost, proto mohou být tato značení zaměňována.

Mějme oblasti $\mathbf{P}$ a $\mathbf{P}'$, respektive značení f a f' a značku α. Přesun značky α z oblasti $\mathbf{P}$ do $\mathbf{P}'$ nazýváme α-expanzí. Navíc musí pro libovolnou značku $l \neq \alpha$ platit $\mathcal{P}_\alpha \subset \mathcal{P'}_\alpha$ a $\mathcal{P}_l \subset \mathcal{P'}_l$. Jinými slovy α-expanze umožňuje libovolné množině pixelů obrazu změnit svoji značku na α.

1. nahodne znaceni f
2. uspech := 0
3. **for** each $\alpha \in \mathcal{L}$
   3.1. nalezni znaceni $\hat{f} = \mathrm{argmin} E(f')$ spolecne s˜f' behem jedne $\alpha-$ expanze f
   3.2. **if** $(E(\hat{f}) < E(f))$ then $f := \hat{f}$, uspech := 1
4. **if** (uspech == 1) then **goto** 2
5. **return** f

Výpis 1: Pseudo kód algoritmu α-expanze

Pseudokódem si nastíníme algoritmus. Samostatné spuštění kroků 3.1-3.2 budeme nazývat iterací a vykonání kroků 2-4 budeme nazývat cyklus. Algoritmus začíná náhodným značením. V každém cyklu pro každou značku vykoná iteraci. V každém cyklu je tedy vykonáno $|\mathcal{L}|$ iterací. Pokud je v jakékoliv iteraci nalezeno jasně lepší značení, je cyklus označen jako úspěšný. Jakmile je cyklus označen jako úspěšný, algoritmus končí a nehledá se žádné další lepší značení.

První důležitou vlastností je, že algoritmus má konečný počet kroků. Je tedy zaručeno, že bude ukončen. Experimentálně jsem ověřil, že algoritmus končí v rámci jednotek cyklů. Většina vylepšení se děje v prvním cyklu. Druhou důležitou vlastností je, že jakmile je algoritmus ukončen, je na výstupu takové značení, které je v mezích globálního minima E podle [21]

$$E(f^*) \leq E(f) \leq 2k \cdot E(f^*), \tag{12}$$

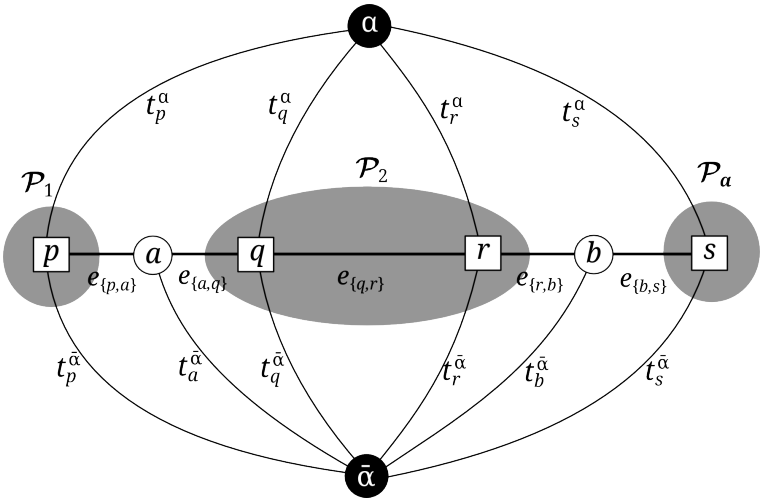
kde f^* značí globální minimum a

$$k = \frac{max\{V(\alpha,\beta) : \alpha \neq \beta\}}{min\{V(\alpha,\beta) : \alpha \neq \beta\}}.$$

3.3.3 Optimální expanze

Máme-li na vstupu algoritmu značení f (oblasti $\mathbf{P}$) a značku α, přejeme si najít značení $\hat{f}$, které pomocí jedné α-expanze minimalizuje energii E. Ukážeme si způsob, jak tento problém vyřešit. Navíc nesmíme stále zapomínat na podmínku, že každá funkce $V_{\{p,q\}}$ musí splňovat podmínky metriky. Výpočet je založen na znalosti, že značení odpovídá minimálnímu řezu grafu $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. Abychom mohli provést řez, musíme graf nejprve sestrojit.

Konstrukce grafu Struktura grafu je určena aktuálním dělením $\mathbf{P}$ a značkou α. Takový graf budu značit $\mathcal{G}_\alpha$. Na obrázku č. 3.3 je ukázka struktury grafu pro jednorozměrný obrázek. Množina vrcholů, stejně jako v předchozí kapitole, obsahuje vrcholy reprezentující



Obrázek 3.3: Ukázka grafu $\mathcal{G}_\alpha$ v jednorozměrném případě. $\mathcal{P} = \{p, q, r, s\}$ je množina pixelů obrazu. Dále jdou vidět oblasti $\mathcal{P}_1 = \{p\}$, $\mathcal{P}_2 = \{q, r\}$ a $\mathcal{P}_\alpha = \{s\}$. Celá množina dělení je pak $\mathbf{P} = \{\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_\alpha\}$. Mezi sousedními pixely, kde každý leží v jiné oblasti jsou přidány pomocné uzly $a = a_{\{p,q\}}$ a $b = a_{\{r,s\}}$.

pixely obrazu a dva terminály, tentokrát označené jako α a $\overline{\alpha}$. Dále jsou mezi sousedními pixely $\{p, q\} \in \mathcal{N}$, kde každý leží v jiné oblasti, přidány pomocné vrcholy $a_{\{p,q\}}$. Celá množina vrcholů se pak dá zapsat jako

$$\mathcal{V}_\alpha = \{\alpha, \overline{\alpha}, \mathcal{P}, \bigcup_{\{p,q\}\in\mathcal{N};f_p\neq f_q} a_{\{p,q\}}\}.$$

Každý vrchol reprezentující pixel obrazu je spojen s terminálem α t-hranou t_p^α a s terminálem $\overline{\alpha}$ t-hranou $t_p^{\overline{\alpha}}$. Každé dva sousední pixely $\{p, q\} \in \mathcal{N}$, které leží v jedné oblasti jsou spojeny n-hranou $e_{\{p,q\}}$. Mezi dvěma sousedními vrcholy, které neleží v jedné oblasti ($f_p \neq f_q$), respektive mezi mezi dvěma sousedními vrcholy a pomocným vrcholem, je vytvořena trojice hran $\mathcal{E}_{\{p,q\}} = \{e_{\{p,a\}}, e_{\{a,q\}}, t_a^{\overline{\alpha}}\}$. Celá množina hran se ve výsledku dá zapsat jako

$$\mathcal{E}_\alpha = \{\bigcup_{p\in\mathcal{P}} \{t_p^\alpha, t_p^{\overline{\alpha}}\}, \bigcup_{\{p,q\}\in\mathcal{N};f_p\neq f_q} \mathcal{E}_{\{p,q\}}, \bigcup_{\{p,q\}\in\mathcal{N};f_p=f_q} e_{\{p,q\}}\}.$$

Jednotlivé ceny hran jsou uvedeny v tabulce č. 2.

Stejně jako bylo řečeno v minulé kapitole, řez $\mathcal{C}$ na grafu $\mathcal{G}$ musí, pro každý vrchol reprezentující pixel, oddělit právě jednu t-hranu. To vede na značení $f^\mathcal{C}$, které odpovídá řezu $\mathcal{C}$

hrana	cena	pro
$t_p^{\overline{\alpha}}$	∞	$p \in \mathcal{P}_\alpha$
$t_p^{\overline{\alpha}}$	$D_p(f_p)$	$p \notin \mathcal{P}_\alpha$
t_p^{α}	$D_p(\alpha)$	$p \in \mathcal{P}$
$e_{\{p,a\}}$	$V_{\{p,q\}}(f_p, \alpha)$	$\{p,q\} \in \mathcal{N}, f_p \neq f_q$
$e_{\{a,q\}}$	$V_{\{p,q\}}(\alpha, f_q)$	
$t_a^{\overline{\alpha}}$	$V_{\{p,q\}}(f_p, f_q)$	
$e_{\{p,q\}}$	$V_{\{p,q\}}(f_p, \alpha)$	$\{p,q\} \in \mathcal{N}, f_p = f_q$

Tabulka 2: Ceny hran grafu α-expanze

$$f_p^{\mathcal{C}} = \begin{cases} \alpha & pokud \quad t_p^{\alpha} \in \mathcal{C} \\ f_p & pokud \quad t_p^{\overline{\alpha}} \in \mathcal{C} \end{cases} \qquad \forall p \in \mathcal{P}. \tag{13}$$

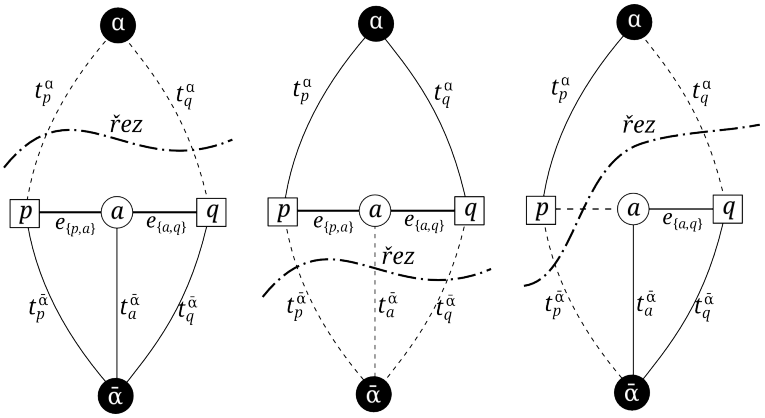
Značení a řez grafu Nyní se ještě podívejme na množinu hran $\mathcal{E}_{\{p,q\}}$, ležících mezi sousedními pixely $\{p,q\} \in \mathcal{N}$, které neleží v jedné oblasti ($f_p \neq f_q$). U těchto hran je několik způsobů jak provést řez. A to i přesto, že pár oddělených t-hran z pixelů p a q je pevně daný. Proto jsou pro oddělení hran $\mathcal{E}_{\{p,q\}}$ v závislosti na dělených t-hranách pro minimální řez $\mathcal{C}$ dány následující pravidla [3]:

$$\begin{array}{llllll} a) & Pokud & t_p^{\alpha}, t_q^{\alpha} \in \mathcal{C} & pak & \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = \emptyset. \\ b) & Pokud & t_p^{\overline{\alpha}}, t_q^{\overline{\alpha}} \in \mathcal{C} & pak & \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = t_a^{\overline{\alpha}}. \\ c) & Pokud & t_p^{\overline{\alpha}}, t_q^{\alpha} \in \mathcal{C} & pak & \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = e_{\{p,a\}}. \\ d) & Pokud & t_p^{\alpha}, t_q^{\overline{\alpha}} \in \mathcal{C} & pak & \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = e_{\{a,q\}}. \end{array} \tag{14}$$

Pravidlo (a) vyplývá z faktu, že žádná podmnožina $\mathcal{C}$ není řezem. Další pravidla plynou z minimální ceny řezu $|\mathcal{C}|$ a faktu, že ceny hran $|\mathcal{E}_{\{p,a\}}|$, $|\mathcal{E}_{\{a,q\}}|$ a $|t_a^{\overline{\alpha}}|$ splňují podmínku trojúhelníkové nerovnosti (11). Takže řez na jedné z nich je levnější než na ostatních dvou dohromady. Tyto vlastnosti je možné vidět ještě na obrázku č. 3.4.

Na začátku této podkapitoly bylo zmíněno, že výpočet algoritmu je založen na znalosti, že značení odpovídá minimálnímu řezu grafu $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. Toto tvrzení z [3] si lépe uvedeme následující větou.

Věta 3.3 *Nechť je zkonstruován graf $\mathcal{G}_\alpha$ a dáno značení f a značka α. Potom elementární řezy na $\mathcal{G}_\alpha$ odpovídají jedna ku jedné značení v rámci jedné α-expanze na f. Navíc pro jakýkoliv elementární řez $\mathcal{C}$ platí $\mathcal{C} = E(f^{\mathcal{C}})$.*

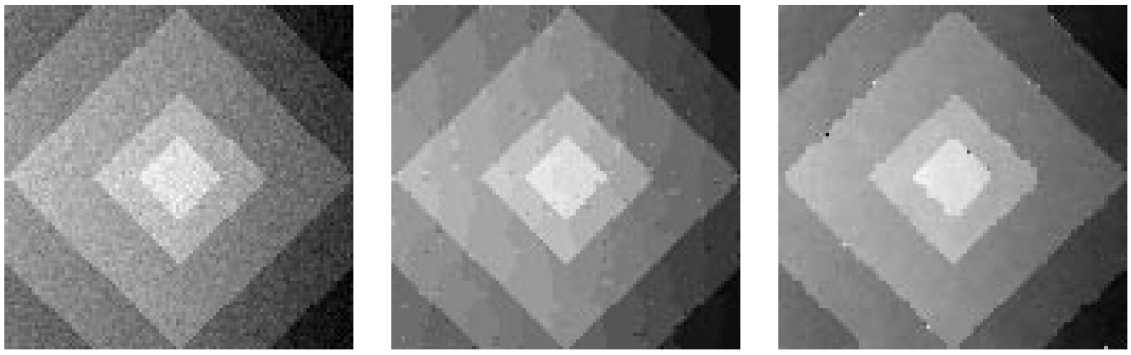


Obrázek 3.4: Vlastnosti minimálního řezu $\mathcal{C}$ na grafu $\mathcal{G}_\alpha$ pro dva sousedící pixely $p, q \in \mathcal{N}$, takové, že $f_p \neq f_q$. První obrázek ukazuje vlastnost 14(a), druhý 14(b) a třetí ukazuje vlastnosti 14(c,d).

Důkaz věty 3.3 je uveden v [3].

3.3.4 Rekonstrukce obrazu

Závěrem popisu algoritmu α-expanze uvedu praktickou ukázku jeho použití. Jedna z možností použití je i rekonstrukce obrazu. Zde je při výpočtu jako funkce V použita euklidovská vzdálenost $V_{\{p,q\}}(f_p, f_q) = min(K, ||f_p - f_q||)$.



Obrázek 3.5: Ukázka rekonstrukce obrazu pomocí algorimtu α-expanze. První obrázek je původní obrázek poškozený šumem, druhý je obraz po rekonstrukci algoritmem α-expanze a třetí je pro srovnání obraz po rekonstrukci algoritmem simulated annealing – simulované žíhání. Zdroj: [3]

3.4 Minimalizace energie s cenami značek

Algoritmus α-expanze měl díky své obecnosti, efektivitě a rychlosti významný dopad na digitální zpracování obrazu [2]. Algoritmus má však drobnou nevýhodu, kterou je pevný počet použitých značek. Na výstupu mohou být tedy i značky, které jsou určitým způsobem nadbytečné. Proto A. Delong et al. vymysleli a popsali v článku [2] metodu, kde má každá značka navíc svou cenu. V průběhu výpočtu se tedy navíc optimalizují i ceny značek a na výstupu jich není více, než je potřeba. Zjednodušeně řečeno, pokud má některá značka příliš vysokou cenu, není dále ve výpočtu použita.

Cenu všech značek zapíšeme jako součet značek, které jsou použity ve značení f

$$\sum_{l \in \mathcal{L}} h_l \cdot \delta_l(f), \tag{15}$$

kde h_l je nezáporná cena značky l a funkce $\delta_l(\cdot)$ určuje, zda je značka použita ve značení f

$$\delta_l(f) \stackrel{def}{=} \begin{cases} 1 & \exists p : f_p \in l \\ 0 & \text{jinak} \end{cases} . \tag{16}$$

Výsledná funkce $E(f)$ je definována jako

$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \sum_{\{p,q\} \in \mathcal{N}} V_{\{p,q\}}(f_p, f_q) + \sum_{l \in \mathcal{L}} h_l \cdot \delta_l(f). \tag{17}$$

Cena nemusí být použita jen přímo pro značky, ale taky pro celou podmnožinu značek h_L. Tomu se více věnuje následující kapitola.

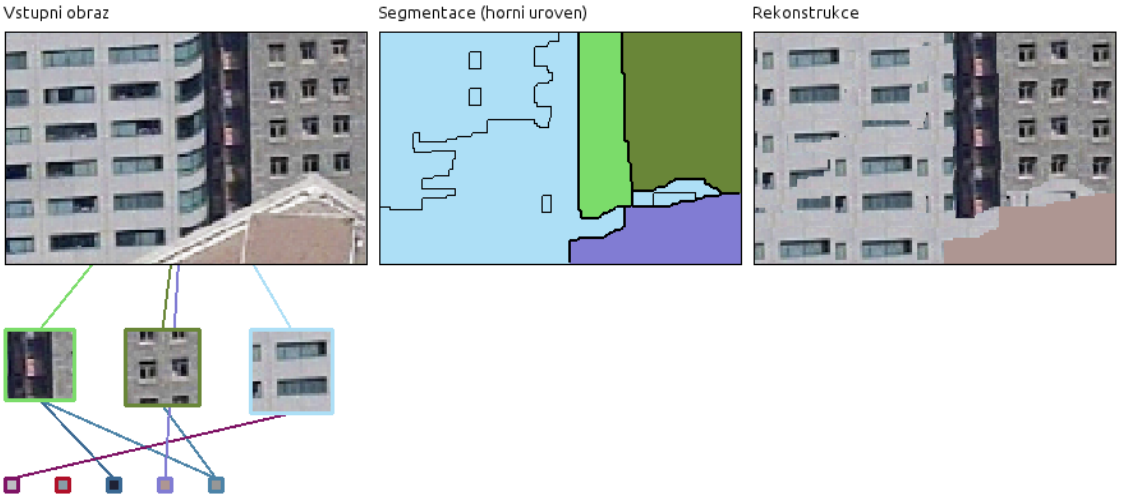
4 Rekurzivní výpočet MDL pomocí grafových řezů

4.1 Úvod

V této kapitole už se dostávám k hlavnímu tématu práce. Tím je segmentační algoritmus popsaný v článku [1] "Recursive MDL via Graph Cuts: Application to Segmentation". Na úvod se pokusím velmi stručně popsat princip a záměr algoritmu. Po úvodu však bude uveden přesnější (více matematický) popis.

Cílem autorů článku bylo vytvořit takový algoritmus, který v obrazu detekuje oblasti, ve kterých se opakuje nějaký vzor. Aby mohly být oblasti detekovány, musí algoritmus jednotlivé vzory znát předem. Jelikož se vzory často neopakují pravidelně, je k nim přidružena sada jednoduchých transformací.

Pro lepší představu se podívejme na obrázek č. 4.1. Na prvním obrázku jsou vidět



Obrázek 4.1: Ukázka reprezentace pomocí záplat

tři oblasti, kde v každé z nich má dům jinou fasádu a jiná okna. Právě pro detekci těchto oblastí je algoritmus vhodný. Pro každou z oblastí byl algoritmu předložen jeden vzor a několik transformací. Vzory jsou vidět pod vstupním obrázkem. Velice zjednodušeně řečeno, algoritmus postupně vezme každý vzor a pod všemi jeho známými transformacemi jej přikládá k obrázku. V každém místě obrázku si algoritmus vypočítá, jak který vzor odpovídá obrázku. To by však ještě nestačilo. Pro oblast, kde by se správně měl opakovat jeden vzor, by mohla nastat situace, kde by v určitém místě lépe pasoval vzor jiné oblasti. Např. díky většímu množství dostupných transformací. Proto je každý přechod z jednoho vzoru na druhý penalizován. Algoritmus potom kombinuje odlišnost vzorů od obrázku spolu s penalizací za přechod a zvolí ten vzor, který mu lépe vyhovuje. Na druhém obrázku je vidět výsledek segmentace. Jednotlivé barvy určují, jaký vzor byl

v daném místě vybrán jako nejlépe vyhovující. Na posledním obrázku je vidět, jak lze zpětně pomocí vypočítané segmentace a použití původních vzorů obrázek rekonstruovat.

Autoři označují v textu vzor spolu s transformacemi jedním slovem jako záplata (patch). Já se tedy tohoto značení budu také držet. Pro záplatu platí ještě pravidlo, že by měla být čtvercového tvaru.

Použití záplat k segmentaci vede na tzv. záplatový model obrázku (patch-based). Algoritmus nemůže mít jistotu, že na vstup dostane přesný počet vhodných záplat. Kdyby algoritmus použil všechny záplaty na vstupu, mohl by být výsledný model zbytečně složitý. Proto je ještě při výpočtu segmentace započítána i složitost modelu. Ve výsledku mohou být některé záplaty tedy vyřazeny.

Reprezentace obrázku pomocí rekurze a záplat

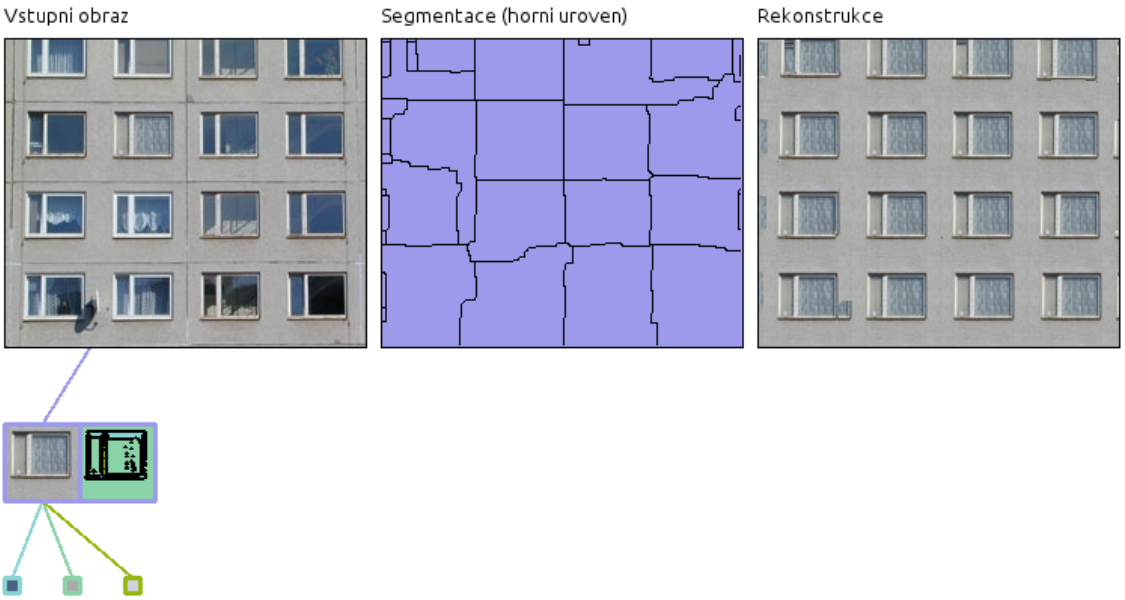
Dosud jsem hovořil pouze o segmentaci vstupního obrazu. Když si však uvědomíme, že záplata je také obraz, můžeme na ní aplikovat stejnou myšlenku. Takže ji segmentujeme dalšími záplatami, dokud nenarazíme na samotné pixely obrázku. Tímto rekurzivním procesem vznikají mezi záplatami vazby. Vyústěním všech záplat a vazeb je pak orientovaný acyklický graf (RDAG - rooted directed acyclic graph). Kořenem grafu je původní obrázek. V každé úrovni grafu je vyvolána segmentace a algoritmus tedy umožňuje tzv. vícenásobnou segmentaci.

Pro segmentaci obrazu, neboli nalezení jeho záplatového modelu, je využito principu hledání minimální délky popisu (MDL - minimum description length). Aby mohl být tento princip použit, je potřeba dostat patřičný počet bitů jako výsledek, jak která záplata odpovídá obrázku. Stejně tak to platí pro penalizaci přechodu mezi záplatami a složitost záplatového modelu. Princip MDL pak vypočítá nejmenší možný počet bitů a tím tedy dostaneme výslednou segmentaci. Podle autorů odpovídá princip minimální délky popisu minimalizaci energie s více značkami. Toto téma jsem nastínil už v předchozí kapitole. K výpočtu segmentace tedy budou použity grafové řezy, a proto jim byla v předchozí kapitole věnována náležitá pozornost.

Algoritmus hierarchického splynutí

Aby bylo možné vypočítat reprezentaci obrázku pomocí minimální délky popisu, je nutné provádět výpočet souběžně na všech možných částech obrázku, na všech možných transformacích a na všech možných záplatách. Podle autorů na řešení tohoto problému není dosud známa žádná vhodná efektivní metoda. Ani algoritmus α-expanze, o kterém jsem hovořil v kapitole 3.3, není v tomto případě dostačující.

Proto autoři navrhli nový algoritmus hierarchického splynutí (HF - hierarchical fusion). Ten v podstatě řeší daný problém po částech. Nejprve spočítá MDL jednotlivým záplatám pro jejich transformace a pak teprve výslednou minimální délku popisu obrázku z MDL



Obrázek 4.2: Na obrázku jde vidět, jak lze pomocí jedné záplaty a několika transformací docílit segmentace celého obrázku. Změna transformace je naznačena tenkou čarou.

jednotlivých záplat. Odtud je také název hierarchického splynutí. Nejdříve splynou jednotlivé transformace a teprve poté celé záplaty. Tím se algoritmus vyhne problémům s lokálním minimem, které má metoda v článku [2].

4.2 Aplikace rekurze a minimální délky popisu

Dostávám se k podrobnějšímu vysvětlení teorie, na níž stojí celý algoritmus. Cílem je nalézt minimální délku popisu obrázku a tím dosáhnout dobré segmentace. Přistoupím k samotnému návrhu rekurzivní energie, která počítá minimální délku popisu s ohledem na množinu záplat. Dále přesněji ukážu, jak je v každém rekurzivním kroku odpovídající záplata rozdělena do souvislých oblastí a popsána ostatními záplatami. Výsledkem bude, jak jsem již zmínil, orientovaný acyklický graf.

4.2.1 Energie rekurze s využitím minimální délky popisu

Obrázek označme I, jednotlivé záplaty $I^1, I^2, \mathrm{atd.}$ a celou množinu záplat jako $\mathcal{S} = \{I^1, I^2, ...\}$. Dále nechť $f : \Omega(I) \to \mathcal{L}$ je značení (labeling), které každému pixelu obrázku I přiřazuje diskrétní značku, která jednoznačně určuje záplatu podléhající nějaké transformaci. Kódovací funkce E_{enc} je zvolena podle dané aplikace. V tomto případě se jedná o kódování, které má vyvolat dobrou segmentaci obrazu. Podrobněji si ji popíšeme v další kapitole. Délka popisu obrázku I s ohledem na množinu záplat $\mathcal{S}$ se pak vyjadřuje

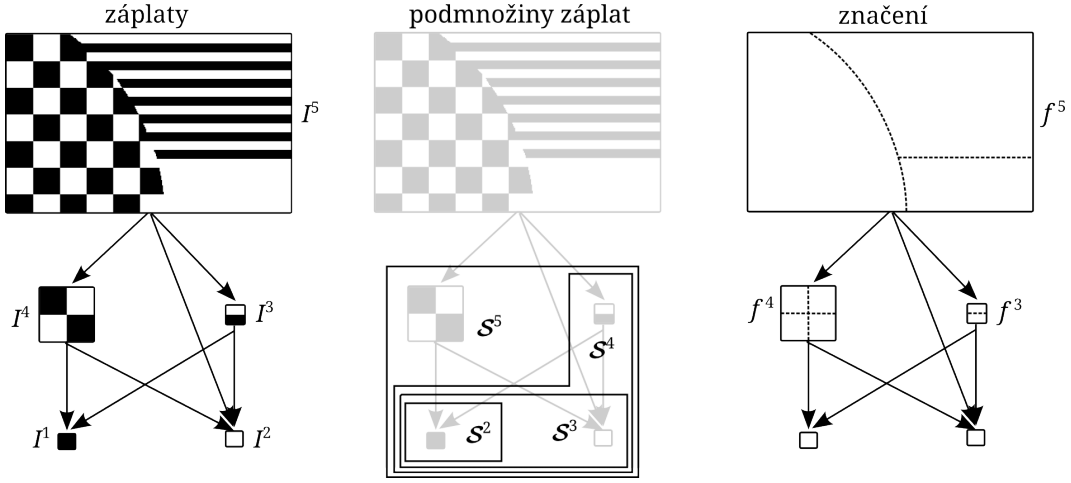
rekurzivně jako

$$\overset{\circ}{E}(\mathcal{S};I) = \min_f E_{\text{enc}}(f;\mathcal{S},I) + \min_{I'\in\mathcal{S}} \overset{\circ}{E}\,(\mathcal{S}\backslash\{I'\};I') + \lg|S|. \tag{18}$$

První část v rovnici (18) $\min_f E_{\text{enc}}(f;\mathcal{S},I)$ minimalizuje délku popisu funkce E_{enc}, která kóduje obrázek I použitím záplat z množiny $\mathcal{S}$. Druhá část rovnice (18) $\min_{I'\in\mathcal{S}} \overset{\circ}{E}\,(\mathcal{S}\backslash\{I'\};I')$ rekurzivně definuje celkovou délku popisu množiny záplat $\mathcal{S}$. Minimalizace probíhá postupně na všech záplatách, kdy se vždy jedna z množiny $\mathcal{S}$ odebere a zakóduje se zbývajícími záplatami. Poslední část rovnice (18) $\lg|S|$ udává počet bitů potřebných k zakódování následující záplaty I', která má být zakódována. Celá rekurze končí v základním případě

$$\overset{\circ}{E}(\{\};I) = E_{\text{direct}}(I), \tag{19}$$

kde musí být záplata I' kódována přímo barvou (úroveň pixelů obrazu). Pokud je použito 24 bitů pro barvu, pak je $E_{\text{direct}}(I) = 24|\Omega(I)|$. Na obrázku 4.3 lze názorně vidět jak reprezentace pomocí záplat vede na RDAG.



Obrázek 4.3: Ukázka kódování obrázku pomocí záplat vedoucí na RDAG. V každém kroku rekurze k je pomocí podmnožiny záplat $\mathcal{S}^k = \{I^1,\ldots,I^{k-1}\}$ nalezeno značení f^k pro záplatu I^k.

Abychom mohli nalézt minimální délku popisu obrázku I, je zapotřebí minimalizovat rekurzivní energii $\overset{\circ}{E}(\mathcal{S};I)$ na všech možných podmnožinách záplat $\mathcal{S}$. Problém lze zapsat vztahem

$$E(I) = \min_{\mathcal{S}} \overset{\circ}{E}(\mathcal{S};I) + 2\lg|\mathcal{S}| + 1. \tag{20}$$

Druhá část rovnice (20) $2\lg|\mathcal{S}| + 1$ značí potřebný počet bitů pro identifikaci množiny záplat $\mathcal{S}$. Rekurzivní energie $\overset{\circ}{E}(\mathcal{S}; I)$ je navržena tak, aby nastavila množiny záplat $\mathcal{S}$, aniž by obsahovaly nadbytečné záplaty. Na tomto místě algoritmus naráží na výpočetní výkon počítačů. Procházení všech možných množin záplat $\mathcal{S}$ je nezvladatelné. Proto potřebné záplaty musíme algoritmu dodat jiným způsobem.

Nejjednodušší možností je záplaty z obrázku získat ručně. Další možností je použít nějaký automatický algoritmus. Autoři článku [1] se odkazují na několik algoritmů, které se podobnou problematikou zabývají. Jedním je využití kompresního algoritmu pro tvorbu kondenzovaného modelu obrázku (condensed epitome) v článcích [8, 9]. Dalším je hledání opakujících se vzorů v obrázku. Tímto tématem se zabývají články [10, 11, 12]. Konkrétní řešení nalezení potřebných záplat však autoři [1] neuvádějí.

4.2.2 Kódovací funkce $E_{\text{enc}}(f; \mathcal{S}, I)$

Dostávám se k popisu samotné funkce $E_{\text{enc}}(f; \mathcal{S}, I)$ pro kódování obrázku I s ohledem na množinu záplat $\mathcal{S}$. Předpokladem je obrázek čtvercového tvaru, tedy o rozměrech $n \times n$. Celý vztah kódovací funkce je definován jako

$$E_{\text{enc}}(f; \mathcal{S}, I) = 2 + 2\lg n + \min\Big\{E_{\text{direct}}(I), \tag{21}$$

$$E_{\text{data}}(f; \mathcal{S}, I) + E_{\text{partition}}(f; \mathcal{S})\Big\}. \tag{22}$$

O značení $f : \Omega(I) \to \mathcal{L}$ jsem už hovořil. Každému pixelu obrázku I přiřazuje nějakou značku z množiny všech značek $\mathcal{L}$. Každou značku tvoří dvojice (e, t). Proměnná e identifikuje nějakou záplatu I^e. t je indexem nějakého transformačního pravidla dostupného pro záplatu I^e. Množinu takových pravidel označme $T(e)$. Platí tedy $t \in T(e)$. Index t tedy určuje transformační pravidlo $\Omega(I) \to \Omega(I^e)$, které mapuje pixely ze souřadného systému obrázku do souřadného systému záplaty. Pro každou záplatu I^e je zvláštní množina transformačních pravidel, a proto celkový počet značek $|\mathcal{L}| = \sum_e T(e)$.

Funkce E_{enc} rozhoduje, jestli se obrázek I popíše přímo použitím barvy pro pixel ($E_{\text{direct}}(I)$ v rovnici (21)) nebo se obrázek I rozdělí na segmenty (rovnice (22)) pomocí optimálního značení f. Každý segment je jednoznačně určen záplatou z množiny $\mathcal{S}$. K rozlišení, kterou z možností funkce použila, ji stačí jeden bit. Zbývajících $1 + 2\lg n$ bitů slouží k zakódování velikosti obrázku I.

V případě, že je použita druhá možnost, tedy že je obrázek dělen na segmenty, vybírá funkce E_{enc} optimální značení f. Optimálním se rozumí takové značení, které dává do rovnováhy složitost dělení obrázku a kvalitu, jak daná záplata odpovídá obrázku.

Energie ceny dat Jak příslušná záplata odpovídá obrázku, budu dále nazývat energií ceny dat. Pokud máme značení f, které přiřazuje každému pixelu obrázku nějaký

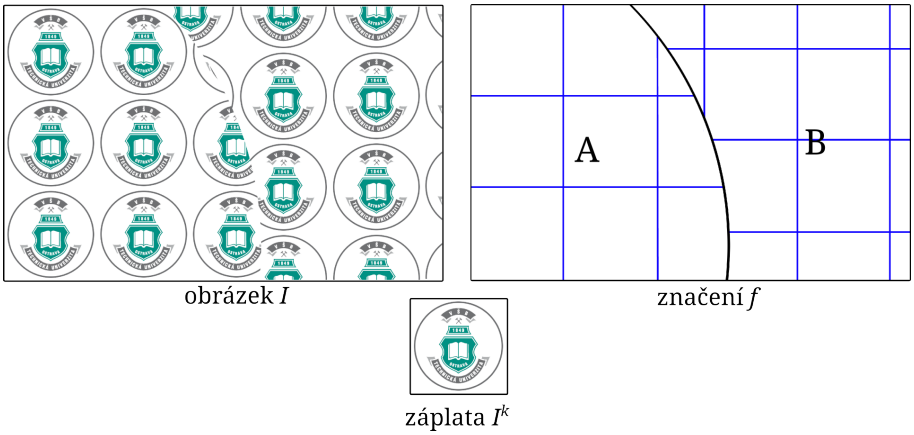
pixel záplaty, tak energie ceny dat E_{data} počítá počet dalších potřebných bitů nutných k rekonstrukci původního obrázku s ohledem na hodnoty barvy záplaty.

Rozebereme si to ještě detailněji. Nechť má každý pixel obrázku $p \in \Omega(I)$ vlastní značku $f_p = (e_p, t_p)$ z množiny všech značek $\mathcal{L}$, kde je e_p indexem záplaty z množiny $\mathcal{S}$ a $t_p \in T(e_p)$ je indexem jednoho transformačního pravidla dostupného pro záplatu I^{e_p}. Dále nechť $I(p)$ značí barvu obrazového pixelu p a $I^{e_p}(t_p(p))$ je barva odpovídajícího pixelu záplaty, kterou určuje značka f_p. Celkový potřebný počet bitů ke kódování obrázku I pomocí záplat $\mathcal{S}$, které se vyskytují ve značení f je dán vztahem

$$E_{\text{data}}(f; \mathcal{S}, I) = \sum_{p \in \Omega(I)} D(I(p) - I^{e_p}(t_p(p))), \tag{23}$$

kde $D()$ udává počet bitů nutných k zakódování každého rozdílu mezi barvou obrázku a záplaty.

Na obrázku 4.4 ukazuji jednu z možných transformací tzv. obkládání (tiling).



Obrázek 4.4: Příklad použití dvou transformačních pravidel pro záplatu. V segmentu A je použito transformační pravidlo obkládání $t_a : (x, y) \rightarrow (x \ mod \ n, y \ mod \ n)$ a v segmentu B je $t_b : (x, y) \rightarrow (x + \frac{n}{2} \ mod \ n, y + \frac{n}{2} \ mod \ n)$. V jednotlivých oblastech jsou použity značky $f_p = (k, t_a)$ a $f_p = (k, t_b)$.

Energie dělení obrázku Energie dělení (členění) obrázku počítá počet bitů nutných k zakódovaní značení f jako takového. Nás zajímá takové značení, které je jednoduché a to ve smyslu, že je prostorově souvislé. Takže s ohledem na množinu párů sousedících pixelů $\mathcal{N}(I)$, musí mít značení f možnost velké komprese.

Složitost dělení je závislá na dvou faktorech. Prvním je počet jedinečných značek a druhým je počet nespojitostí $f_p \neq f_q$ pro libovolné dva sousedící pixely obrázku ($pq \in \mathcal{N}(I)$). Jelikož každá značka f_p obsahuje dvě komponenty (index záplaty a index transformace)

a také platí $e_p \neq e_q \Rightarrow t_p \neq t_q$, tak můžou nastat dva typy nespojitostí. Prvním typem je, že se změní pravidlo transformace, a druhým je, že se změní jak pravidlo transformace, tak záplata. Můžete si toho všimnout na obrázcích 4.1, 4.2.

Celkový počet bitů potřebných k zakódování značení f se pak vyjádří jako

$$E_{\text{partition}}(f;\mathcal{S}) = E_{\text{lookup}}(f;\mathcal{S}) + \sum_{pq\in\mathcal{N}(I)} V(f_p, f_q, f), \tag{24}$$

kde V určuje počet bitů nutných k zakódováni přechodu ze značky f_p na značku f_q. Aby byla délka popisu funkce pro přechody V kratší a abychom si udrželi značení, které využívá méně jedinečných značek (a tím je jednodušší), je zavedena ještě vnitřní energie E_{lookup}. Ta reprezentuje počet bitů navíc, které jsou nutné k zapamatování, které záplaty a transformace jsou použity ve značení f.

Abychom si přesněji vysvětlili, jak funguje funkce V, musíme nejdříve nadefinovat některé nové množiny. K dispozici máme množinu záplat $\mathcal{S}$ a značení f. První množinou nechť je $\hat{\mathcal{S}} = \{i|\exists e_p = i\}$ množina indexů záplat, které jsou aktuálně využity ve značení f. Podobně jako $\hat{\mathcal{S}}$ je definována množina $\hat{T}_i = \{j \in T(i)|\exists f_p = (i,j)\}$ jako množina indexů transformačních pravidel, které záplata I^i současně využívá. Celková cena V se definuje jako počet bitů potřebných k popsání přechodu mezi značkami

$$V(f_p, f_q, f) = \begin{cases} 2 + \lg|\hat{T}_{eq}| + \lg|\hat{\mathcal{S}}| & e_p \neq e_q, t_p \neq t_q \\ 2 + \lg|\hat{T}_{eq}| & e_p = e_q, t_p \neq t_q \\ 1 & \text{jinak.} \end{cases} \tag{25}$$

Zbývá nám nadefinovat energii E_{lookup}. Té je zapotřebí k indexování použitých záplat a transformačních pravidel, protože V je závislá pouze na počtu jedinečných značek, které jsou momentálně použity značením f. E_{lookup} tedy reprezentuje nutný počet bitů k zapsání uvažované tabulky indexů. Je definována jako

$$E_{lookup}(f;\mathcal{S}) = \hat{\mathcal{S}} \lg|\mathcal{S}| + \sum_i |\hat{T}_i| \lg|T_i|. \tag{26}$$

Na tomto místě je důležité si uvědomit, že ceny V jsou závislé na pořadí pixelů. Tedy, že $V(f_p, f_q, f) \neq V(f_q, f_p, f)$. Tím nejsou splněny podmínky nutné pro výpočet α-expanze uvedené v kapitole 3.3 a přímé použití α-expanze pro optimalizaci není možné. Algoritmus α pro optimalizaci využít chceme, proto autoři [1] ke splnění podmínek vymysleli menší úpravu. Ceny V se vypočtou v obou směrech, tedy $V(f_p, f_q, f)$ i $V(f_q, f_p, f)$, a z nich se spočítá průměr. Minimum výsledné energie přitom stále odpovídá té původní.

4.3 Optimalizace rekurzivní energie

Nyní se konkrétněji podíváme na problém minimalizace rekurzivní energie $\overset{\circ}{E}(\mathcal{S};I)$ na všech možných záplatách $\mathcal{S}$ (vztah (20)). Problém je obtížně řešitelný, a to ze dvou hlav-

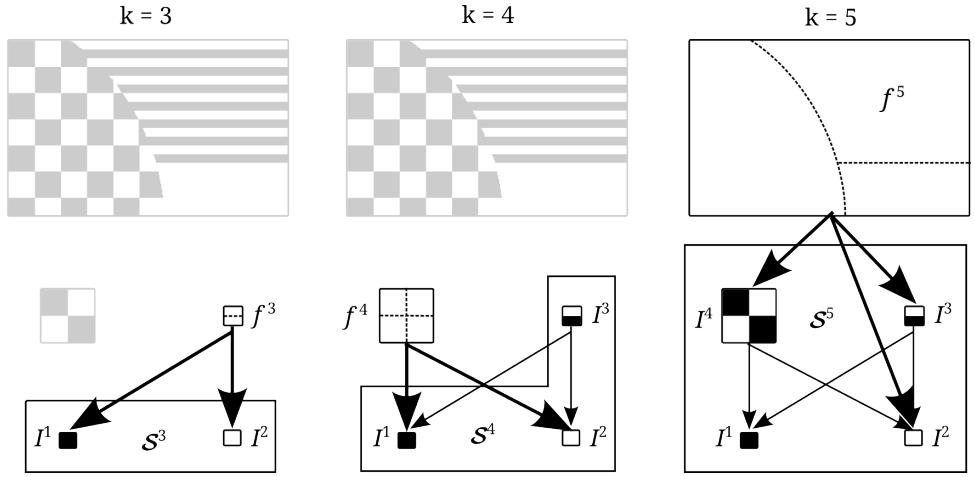
ních důvodů. Za prvé je prakticky velice obtížné procházet všechny možné množiny záplat $\mathcal{S}$ a za druhé klíčový podproblém minimalizace energie E_{enc} uvnitř každého rekurzivního kroku je NP-těžký problém [1]. První problém je řešitelný hltavým přístupem, který si popíšeme v následující části. Podproblém minimalizace E_{enc} se provádí na více značkách (multi-label) s cenami definovanými na příliš velkých podmnožinách značek a ani algoritmus pro optimalizaci v článku [2] není dostačující. Proto je vymyšlen nový algoritmus hierarchického splynutí, který dané energie optimalizuje lépe a popíšeme si ho v kapitole 4.3.2.

4.3.1 Postup konstrukce grafu

Jelikož není možné procházet přes všechny možné podmnožiny záplat, omezujeme se na určitou množinu záplat, které nějakým způsobem z obrázku získáme (manuálně, nějakým automatickým algoritmem). Takovou množinu si označme $\overline{\mathcal{S}}$.

A začneme s hltavým přístupem. Nejprve vzestupně seřadíme všechny záplaty z množiny $\overline{\mathcal{S}}$. Následně postupně tvoříme acyklický orientovaný graf. V každém kroku k přidáme jednu záplatu a vypočteme minimální délku popisu obrázku I^k zakódováním záplat z množiny $\overline{\mathcal{S}}^k = \{I^1, I^2, ..., I^{k-1}\}, \overline{\mathcal{S}}^k \subseteq \overline{\mathcal{S}}$. Předpokladem je, že v kroku k už záplaty $\{I^1, I^2, ..., I^{k-1}\}$ byly zakódovány a spočítány jejich optimální dělení $\{f^1, f^2, ..., f^{k-1}\}$ a tudíž vazby mezi záplatami v množině $\overline{\mathcal{S}}^k$ jsou pevné.

Na obrázku č. 4.5 si můžeme všimnout několika kroků vytváření RDAG.



Obrázek 4.5: Ukázka konstrukce acyklického orientovaného grafu pro kroky $k = 3, 4, 5$.

Samotné vypočtení minimální délky popisu pro záplatu I^k znamená vybrat vhodnou podmnožinu záplat z množiny $\overline{\mathcal{S}}^k$. Výběr vhodné podmnožiny je navíc komplikován tím, že záplata $I^i \in \overline{\mathcal{S}}^k$ může být ke kódování I^k použita jak přímo, tak nepřímo. Avšak cena

záplaty I^i může být použita (zaplacena) nejvýše jednou. Všimněme si toho na obrázku č. 4.5. Značení f^5 používá záplatu I^2 v jedné oblasti přímo a v další (např. přes I^3) nepřímo.

Abychom správně vypočetli vazby mezi záplatami v kroku k, jsou použity energie s cenami podmnožin značek jak je tomu v článku [2]. Tyto ceny reprezentují délku popisu ve vztahu (18), která je počítána rekurzí na libovolné podmnožině množiny $\overline{\mathcal{S}}^k$. Abychom si nadefinovali všechny potřebné podmnožiny značek, začneme s popisem závislostí v hierarchii záplat. Dostáváme se konečně k několikrát zmíněnému orientovanému acyklickému grafu (RDAG) $\mathcal{G} = \{\mathcal{V}^k, \mathcal{A}^k\}$. Stále se držíme kroku k při konstrukci. Vrcholy grafu $\mathcal{V}^k = \{v_1, v_2, ..., v_{k-1}\}$ odpovídají záplatám z množiny $\overline{\mathcal{S}}^k$. Kdykoliv nějaká záplata I^j pro $i < j < k$ při kódování přímo závisí na jiné záplatě I^i, je pro zaznamenání této závislosti použita hrana $(v_j, v_i) \in \mathcal{A}^k$. Dále pro každý vrchol grafu $v_i \in \mathcal{V}^k$ udáváme množinami $\mathcal{X}_i$ $\mathcal{Y}_i$ jeho předchůdce a následovníky v grafu

$$\mathcal{X}_i = \{j \mid v_i \text{ je dosažitelné z } v_j\} \cup \{i\},$$

$$\mathcal{X}_i = \{j \mid v_j \text{ je dosažitelné z } v_i\}.$$

$\mathcal{X}_i$ indexuje všechny záplaty, které, ať už přímo nebo nepřímo, využívají záplatu I^i ke kódování. Naopak $\mathcal{Y}_i$ indexuje všechny záplaty, které jsou, ať už přímo nebo nepřímo, použity pro kódování záplaty I^i.

Abychom si mohli uvést energii, kterou v kroku k optimalizujeme, potřebujeme znát ještě několik věcí. Nechť L_i značí množinu značek z $\mathcal{L}$, které odpovídají záplatám, které používají záplatu I^i pro kódování. Energie musí být definovaná tak, že pokud nějaká záplata $f_p \in L_i$, pak musí být zaplacena cena za kódování záplaty I^i. Takovou cenu budeme značit h_{l_i}. V každém kroku k v naší hltavé konstrukci grafu optimalizujeme energii

$$\tilde{E}^k = \min_f E_{\text{enc}}(f; \overline{\mathcal{S}}^k, I^k) + \sum_{i=1}^{k-1} h_{L_i} \delta_{L_i}(f), \tag{27}$$

kde je funkce δ_{L_i} definovaná na podmnožině L_i jako

$$\delta_{L_i}(f) = \begin{cases} 1 & \exists p : f_p \in L_i \\ 0 & \text{jinak.} \end{cases} \tag{28}$$

Poslední nevysvětlený ve funkci $\tilde{E}^k$ zbývá výpočet ceny značek h_{L_i}. Ten musí být specificky navržen, aby správně počítal závislosti ve struktuře grafu $\mathcal{G}^k$. Je definován jako

$$h_{L_i} = \tilde{E}^i - \sum_{j \in \mathcal{Y}_i} h_{L_i}. \tag{29}$$

Objasněme si proč jsou ceny h_{L_i} odečteny. Jakmile je energie $\tilde{E}^i$ vypočtena, tak v sobě zahrnuje cenu pro kódování všech záplat $I^j, j \in \mathcal{Y}_i$, na kterých je závislá. Už jsem zmínil, že cena může být zaplacena nejvýše jednou, proto všechny ceny $h_{L_j}, j \in \mathcal{Y}_i$ musí být z h_{L_i} odečteny.

Ukážeme si to na krátkém příkladu podle obrázku č. 4.5 a kroku $k = 5$. Podmnožiny značek $\mathcal{X}_i$ vypadají následovně:

$$\begin{aligned}\mathcal{X}_1 &= \{1, 2, 3\},\\ \mathcal{X}_2 &= \{2, 3, 4\},\\ \mathcal{X}_3 &= \{3\},\\ \mathcal{X}_4 &= \{4\}.\end{aligned}$$

Cena $h_{L_1} = \tilde{E}^1$, $h_{L_2} = \tilde{E}^1$. Jelikož však množiny L_1 a L_2 už obsahují záplaty I^3 a I^4 musí být od nich ceny h_{L_1} a h_{L_2} odečteny:

$$\begin{aligned}h_{L_3} &= \tilde{E}^3 - h_{L_1} - h_{L_2},\\ h_{L_4} &= \tilde{E}^4 - h_{L_1} - h_{L_2}.\end{aligned}$$

Celý algoritmus je zachycen v následujícím pseudokódu:

1. **for** k˜= 1..$|\overline{\mathcal{S}}|$
2. $\quad\overline{\mathcal{S}}^k = \overline{\mathcal{S}}^{k-1} \cup \{I^i\}$
3. $\quad\mathcal{V}^{k+1} = \mathcal{V}^k \cup \{v_k\}$
4. $\quad$s˜vyuzitim grafu $\mathcal{G}^k$ nastavit podmnoziny znacek L_i a jejich ceny h_{l_i}
5. $\quad f^k = \operatorname{argmin}_f \tilde{E}^k$ *// minimalizovano HF (kapitola* 4.3.2*)*
6. $\quad$**if** $\tilde{E}^k(f_k)$ nevyuziva $E_{\text{direct}}(I^k)$
7. $\quad\quad\mathcal{A}^{k+1} = \mathcal{A}^k \cup \{(v_k, v_j) \mid \exists f_p^k = (j, \cdot)\}$
8. $\quad$vytvor $\mathcal{X}_k, \mathcal{Y}_k$, aktualizuj $\mathcal{X}_j \;\; \forall j < k$

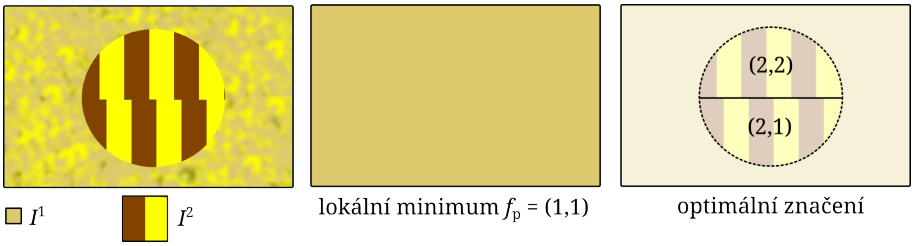
Výpis 2: Pseudo kód hltavého přístupu konstrukce RDAG

4.3.2 Algoritmus hierarchického splynutí

Jelikož využíváme k výpočtu energie (27) ceny značek, nabízí se pro optimalizaci využít algoritmu α-expanze. Avšak pokud jsou ceny značek definovány na velkých podmnožinách, má tato metoda nízkou míru optimalizace a v praxi se zasekává na lokálním minimu[2, 1]. Tento problém je možné postřehnout na obrázku č. 4.7.

Zásadní problém při použití α-expanze je, že žádnou značku (i, t) z množiny L_i se kvůli vysoké ceně H_{L_i} nevyplatí rozšiřovat. Pokud bychom však šířili několik značek souběžně, mohla by klesnout cena dat a tím by se dala kompenzovat vysoká cena značek h_{l_i}. A to je právě základní myšlenka, která vede na algoritmus hierarchického splynutí.

Popíšeme si jej při optimalizaci naší energie $\tilde{E}^k$ v kroku k. Začneme tím, že si nachystáme množinu pod-energií. Jedna energie je pro každou z $k-1$ záplat. Každá pod-energie



Obrázek 4.6: Problém lokálního minima α-expanze s cenami podmnožin značek. Podmnožina L_2 obsahuje značky (2,1) a (2,2). Pokud je použita pouze záplata I^1, cena podmnožiny h_{L_2} je pro obě transformace příliš vysoká a L_2 se tedy nemůže šířit. Proto je nutné šířit všechny transformace současně.

j $(1 \leq j \leq k-1)$ se omezuje na vlastní množinu značek $\mathcal{L}^j = \{(j, \cdot)\}$. Tedy na značky se všemi transformačními pravidly $T(j)$ dostupnými pro záplatu I^j. Připravené pod-energie už můžeme optimalizovat pomocí algoritmu α-expanze [2].

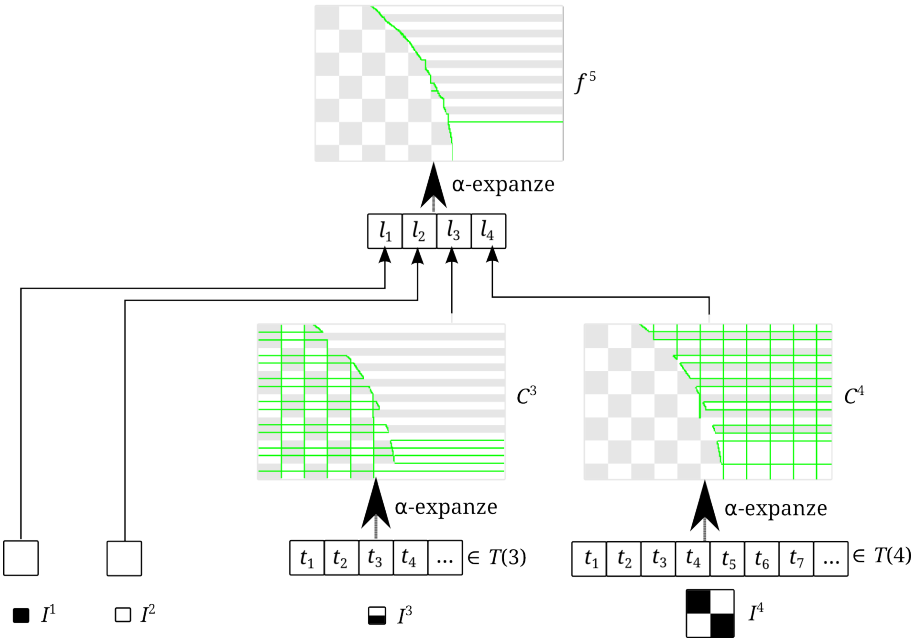
Výsledkem jednotlivých optimalizací je značení $c^j : \Omega(I^k) \rightarrow \mathcal{L}^j$. Každé značení c^j se snaží, pomocí transformací záplaty I^j jejími transformačními pravidly $T(j)$, co nejlépe rekonstruovat barvy pixelů záplaty I^k. Množinu všech takových značení označme $\mathcal{C} = \{c^1, c^2, c^{k-1}\}$.

Na obrázku č. 4.7 si můžeme všimnout, jak je záplata I^5 dokonale rekonstruována značením c^4. Avšak musí za to, v oblastech, které záplatě I^4 příliš neodpovídají, platit vysokou cenu dělení.

Obecně vzato, oblasti záplaty I^k, které místy odpovídají záplatě I^j, budou mít, díky podobné barvě, nízkou cenu dat a díky souvislým oblastem s konstantní transformací i nízkou cenu dělení. Na druhou stranu oblasti, které příliš neodpovídají, budou mít buď vysokou cenu dat nebo dělení.

Když máme spočítáno značení c^j, zavedeme si proměnnou c_p^j, která pro každý pixel $p \in \Omega(I^k)$ označuje transformační pravidlo z $T(j)$. Následně dáme (necháme splynout) všechna značení z množiny $\mathcal{C}$ dohromady do jedné energie. Ta však stále odpovídá energii (27). Každému pixelu definujeme vlastní množinu značek pro splynutí jako $\mathcal{L}_p^{\text{fuse}}$. Tedy přiřazením značky (k, c_p^j) pixelu p, je vybrána reprezentace pixelu p záplatou j podléhající transformaci c_p^j. Optimalizace na množině $\mathcal{L}^{\text{fuse}}$ výrazně zmenšuje velikost každé podmnožiny značek L_i ve vztahu (27). A to z $\sum_{j \in \mathcal{X}_i} |T(j)|$ na $|\mathcal{X}_i|$. Tím se taky vyhneme problémům s lokálním minimem způsobeným cenami h_{L_i}.

Tento algoritmus tedy v kroku k, pomocí rozdělení obrázku a použití transformačních pravidel na podmnožiny záplat z množiny $\overline{\mathcal{S}}^k$, vybere minimální délku popisu záplaty I^k. Celý algoritmus zachycuje ještě následující pseudokód.



Obrázek 4.7: Hierarchické splynutí

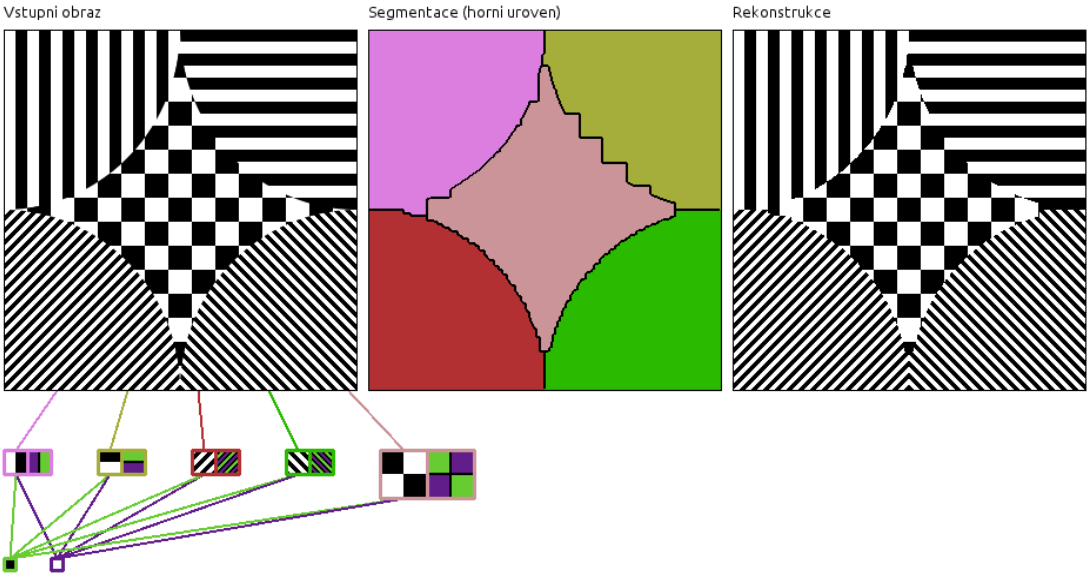
1. **for** j = 1..$k-1$
2. $\mathcal{L}^j = (j, \cdot) \in \mathcal{L}$
3. $c^j = \text{argmin}_c \tilde{E}^k, c : \Omega(I^k) \rightarrow \mathcal{L}^j$ // $\alpha-$*expanze*
4. $\mathcal{L}_p^{\text{fuse}} = \{(j, c_p^j) \,|\, j \in 1 \ldots k\ -1\}\ \forall p$
5. $f^k = \text{argmin}_f \tilde{E}^k,\ f : \Omega(I^k) \rightarrow \mathcal{L}^{\text{fuse}}$

Výpis 3: Pseudo kód hierarchického splynuti v kroku k

5 Experimenty

Po teoretickém popisu metody rekurzivního výpočtu MDL grafovými řezy se dostávám k jejímu praktickému ověření. Jak je výsledná segmentace úspěšná, posoudí nejlépe lidské oko. Proto nabízím k posouzení sadu experimentů, jejichž výpočet byl proveden pomocí vlastní implementace algoritmu. U každého experimentu se vždy vyskytuje vstupní obrázek a obrázek ukazující segmentaci. Barvy v obrázku segmentace značí jednotlivé záplaty. Změna záplaty je zaznamenána tlustší čarou a tenčí čarou jsou pak znázorněny změny transformací v rámci jedné záplaty. Ke vstupnímu obrázku a obrázku se segmentací je většinou přidán graf se záplatami a závislostmi, který vznikl během výpočtu. Dále je přidán obrázek, který by vznikl rekonstrukcí ze záplat a vazeb mezi nimi. Každý výsledek je ještě navíc stručně okomentován.

Experimenty jsem rozdělil do dvou částí. V první části jsou testovány obrázky, které by měly být pro metodu vhodné. V obrázcích jsou tedy oblasti, ve kterých se opakují nějaké textury (záplaty). Ve druhé části je vyzkoušeno, jak metoda zareaguje na dva obrázky, které pro ni příliš vhodné nejsou. V první části je navíc použito dvou typů obrázků. Jedním typem jsou předem připravené, či namalované obrázky s jasně daným očekávaným výstupem. Druhým typem jsou fotografie, které byly vybrány s ohledem na princip metody, a také se u nich očekává určitý výstup.



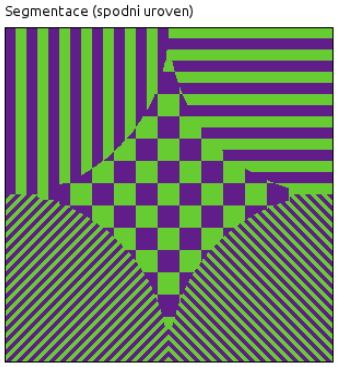
Obrázek 5.1: Uměle vytvořený obrázek, který by měla metoda zvládnout

5.1 Sada obrázků ”vhodných” pro metodu

5.1.1 Ručně připravené obrázky

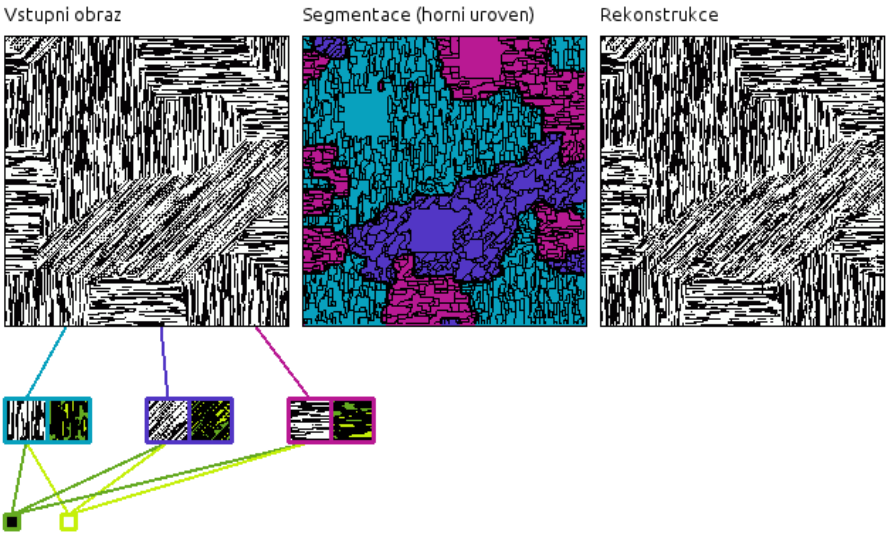
První obrázek (č. 5.1) jsem nakreslil tak, aby byl vhodný pro metodu. Obrázek obsahuje pět oblastí, kde je v každé z nich použita jiná textura. Každá z textur je navíc pravidelná. Jak jsem dříve uvedl, aby mohla být popsanou metodou spočítána segmentace, je potřeba mít na vstupu ještě tzv. záplaty. Tedy části obrázku, které se v něm opakují. V tomto případě bylo pro dobrou segmentaci zapotřebí označit v každé oblasti alespoň jednu záplatu. Jelikož jsou textury pravidelné, tak při vhodně zvolené velikosti záplaty, není pro ni potřeba přidávat další pravidla transformací. Aby mohly být i tyto záplaty segmentovány, bylo nutné označit pixel černé a bílé barvy. Jednotlivé pixely se nacházejí ve spodní části grafu. Zbylé záplaty jsou ve vrstvě nad nimi.

Segmentace je u tohoto obrázku poměrně zdařilá. Pouze v místech, kde se prolínají podobné textury, je hrana segmentace nepřesná. Algoritmus totiž v těchto místech upřednostní cenu z rozdílnosti záplat před cenou jemností přechodu. Vedle každé ze záplat je zobrazena její vlastní segmentace použitím ostatních záplat. Na obrázku č. 5.2 je rekonstruovaný obrázek s použitím segmentací jednotlivých záplat.



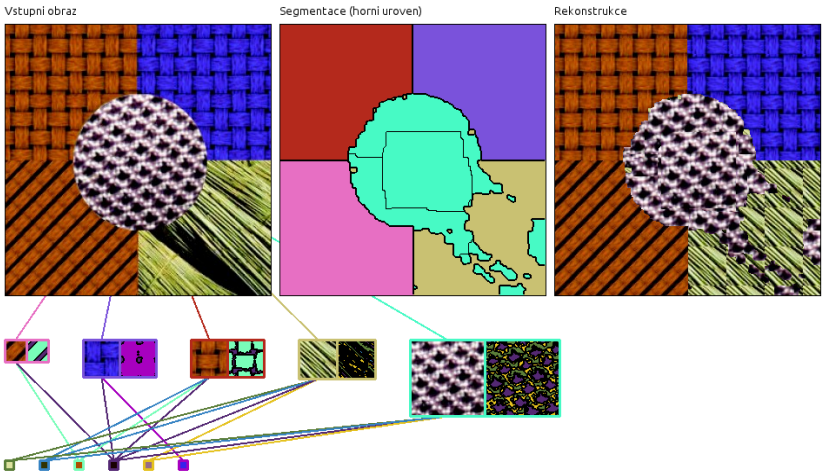
Obrázek 5.2: Rekonstrukce obrazu s použitím dílčích segmentací záplat

Druhý experiment (obrázek č. 5.3) je pro metodu poněkud komplikovanější. V žádné oblasti není vzor, který by se přesně opakoval. Byly proto vybrány 3 záplaty, kde v každé z nich je skupina čar jinak otočená. Aby algoritmus mohl tyto záplaty nalézt i jinde v příslušné oblasti, bylo každé z nich přidáno několik transformačních pravidel. Výsledek segmentace je, řekl bych, docela uspokojivý. Povšimněte si především velkého počtu změn transformačních pravidel (tenké čáry). Právě díky transformacím algoritmus upřednostnil použití záplat před jednotlivými barvami.



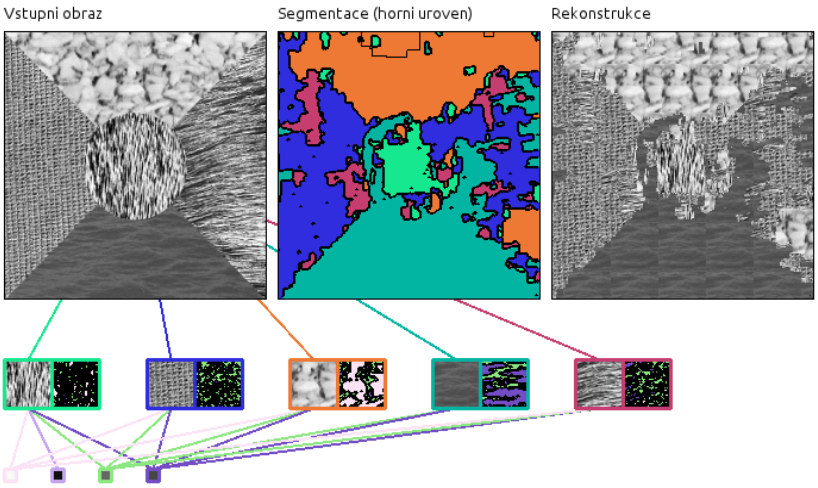
Obrázek 5.3: Uměle vytvořený obrázek s nepravidelnou texturou. Zdroj vstupního obrázku: [22]

V dalším obrázku jsou už složitější vzory a navíc barvy. Modrá část a oranžové části obsahují pravidelný vzor a jejich segmentace vypadá v pořádku. Prostřední oblast a oblast s obilím už příliš pravidelné nejsou. Prostřední kruh je však i tak označen poměrně dobře. V oblasti s obilím je to o něco horší. Očekával bych, že v tmavém místě bude použita přímo tmavá barva. Namísto toho je použita záplata z jiné oblasti. Cena barvy pro pixely byla asi stále větší, než cena použití jiné záplaty.



Obrázek 5.4: Použití barev, pravidelných i nepravidelných vzorů. Zdroj vstupního obrázku: [23]

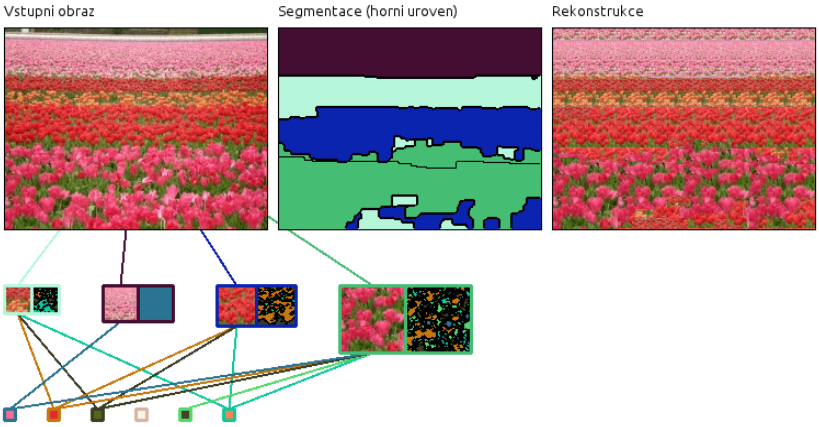
Po třech poměrně zdařilých výsledcích je k dispozici experiment, který se až tak nevyvedl. Ne zrovna dobrý výsledek má nejspíše na svědomí stupnice šedi. Chyby vzniklé při výpočtu, jak která záplata odpovídá obrázku, nejsou tak vysoké jako při použití barvy, a proto je někdy špatně použita záplata i v místech, kde ve skutečnosti není.



Obrázek 5.5: Nepříliš zdařilý výsledek. Zdroj vstupního obrázku: [24]

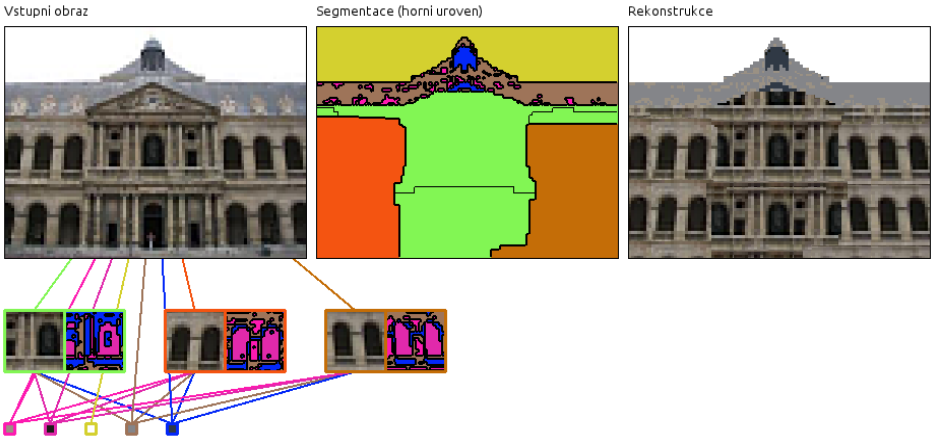
5.1.2 Fotografie

Dostávám se do kategorie s fotografiemi. Jako první jsem vybral fotografii pole s tulipány, kde jsem očekával označení oblastí tulipánů s různou barvou. Výsledek je poměrně dobrý. Pouze v místech kde jsou růžové tulipány opravdu syté, zvolil algoritmus červené.

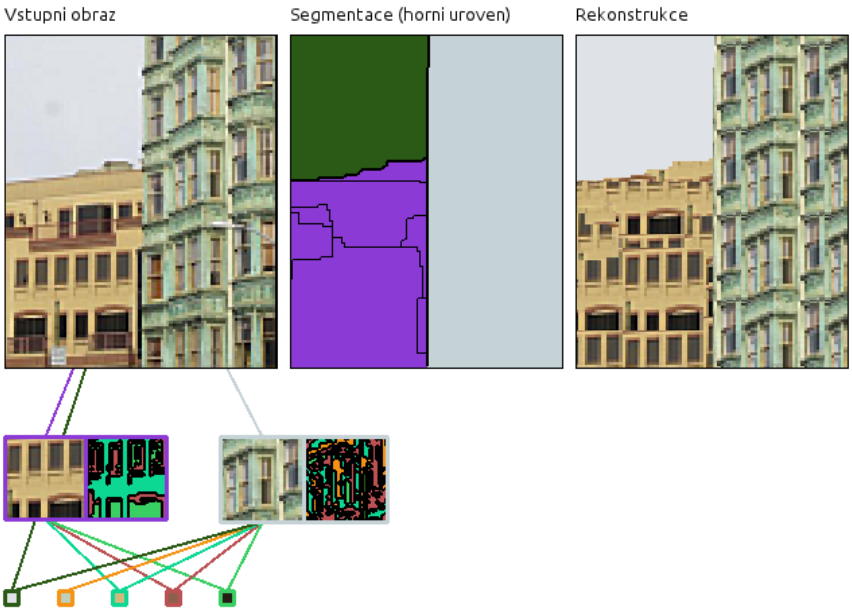


Obrázek 5.6: Segmentace pole s tulipány. Zdroj vstupního obrázku: [25]

Následující dva experimenty považuji za velmi zdařilé. Na prvním se pomocí tří záplat a pěti barev podařilo oddělit levou, prostřední a pravou část domu, střechu a oblohu. Navíc je vidět, jak algoritmus krásně kombinuje záplaty a barvu. Na druhém obrázku můžeme pozorovat, jak lze pomocí dvou záplat a jedné barvy obrázek rekonstruovat.

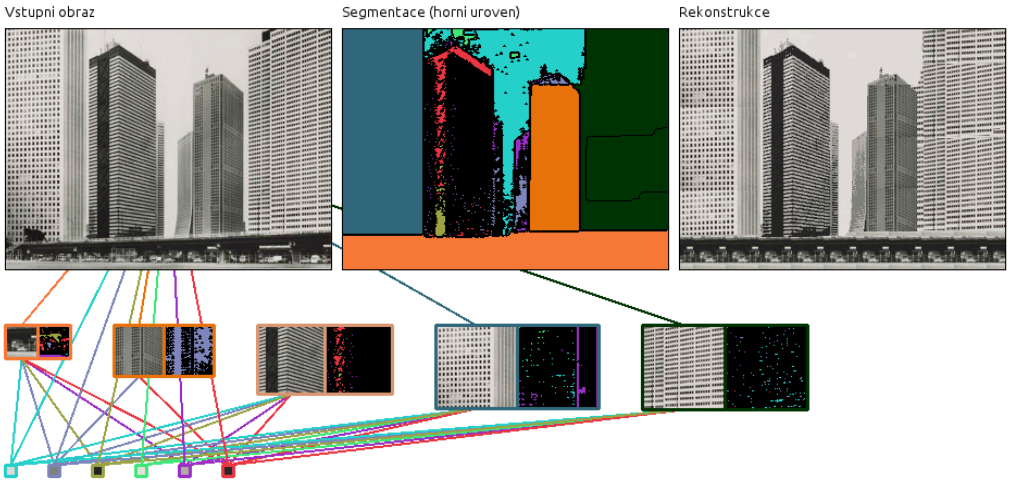


Obrázek 5.7: Výborná segmentace a kombinace záplat a barev. Zdroj vstupního obrázku: [1]



Obrázek 5.8: Zajímavá rekonstrukce obrazu použitím dvou záplat a jedné barvy. Zdroj vstupního obrázku: [1]

Na posledním experimentu ze série fotografií jsem zvolil fotografii mrakodrapů ve stupnici šedi. Segmentace tří mrakodrapů a silnice pod nimi se povedla. U čtvrtého mrakodrapu algoritmus preferoval použití barev před záplatou a jeho segmentace se tedy nepovedla.



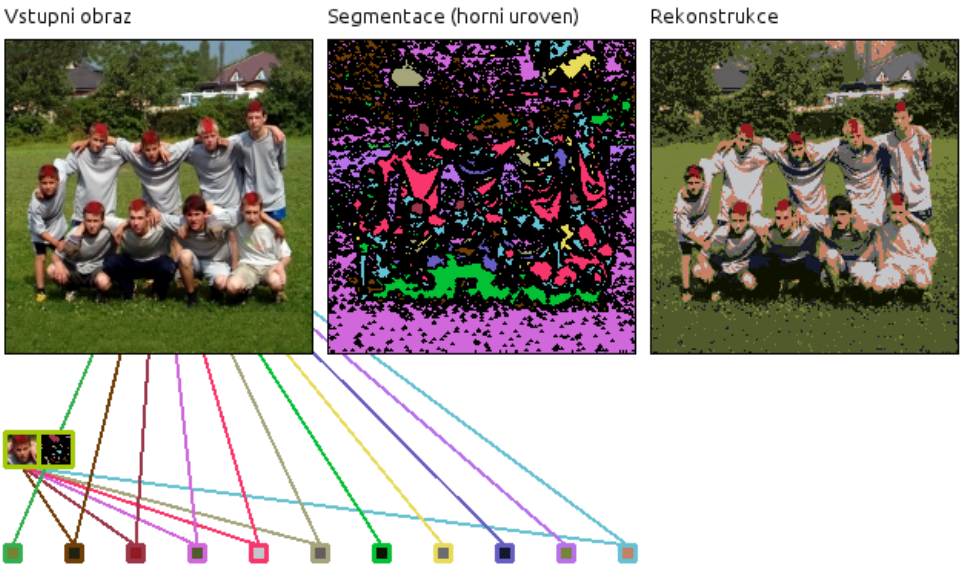
Obrázek 5.9: Segmentace fotografie ve stupnici šedi. Zdroj vstupního obrázku: [26]

5.2 Další experimenty

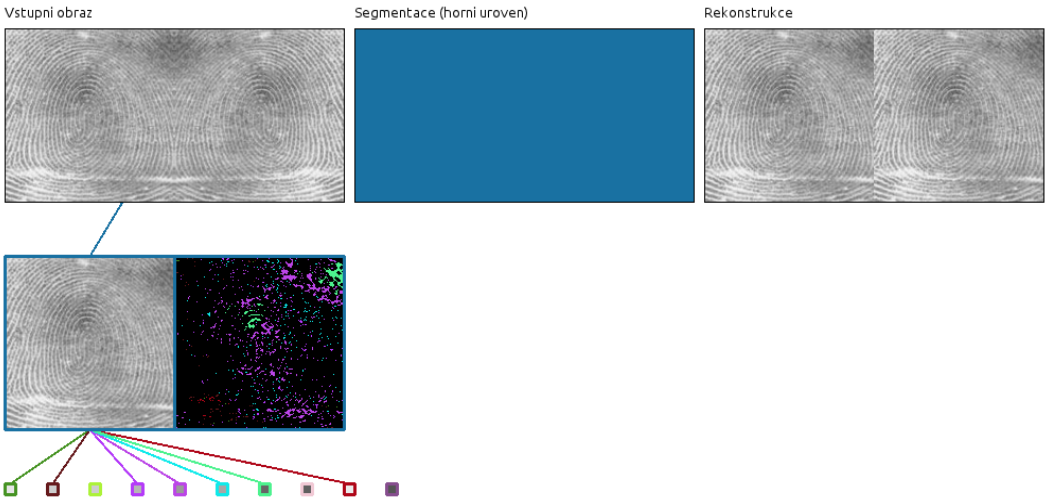
Funkčnost algoritmu jsem ještě otestoval u dvou typu úloh z oblasti biometrie. Obrázky tentokrát nejsou tvořeny oblastmi s opakujícím se vzorem. Nejsou tedy pro algoritmus příliš vhodné. Považuji však za vhodné, v krátkosti ukázat i tento případ.

Na prvním obrázku (č. 5.10) jsem zamýšlel, že by algoritmus mohl detekovat obličeje. Jako záplatu jsem tedy označil jeden obličej a k ní přidal sadu transformačních pravidel, aby "pasovala"i na ostatní obličeje. Pro popis zbytku fotografie jsem na vstup přidal několik barev. Očekávání bohužel naplněno nebylo. Algoritmus totiž preferoval přímé použití barev místo záplaty s obličejem.

V posledním experimentu (obrázek č. 5.11) jsem chtěl vyzkoušet, zda algoritmus od sebe odliší dva otisky prstů. V tomto případě jde konkrétně o jeden otisk prstu, který je zrcadlově obrácený podle osy y. Levý otisk prstu jsem tedy označil jako záplatu a přidal na vstup několik odstínů šedi. Očekával jsem, že první otisk bude správně označen jako otisk a pro druhý otisk algoritmus použije barvy. I v tomto případě mé očekávání nebylo naplněno a algoritmus označil druhý otisk jako podobný tomu prvnímu.



Obrázek 5.10: Pokus o detekci obličejů



Obrázek 5.11: Experiment s otisky prstů. Zdroj vstupního obrázku: [14]

6 Programátorský popis

Z hlediska implementace je tento segmentační algoritmus poměrně rozsáhlý. Důkazem je i velké množství vzorců a vztahů v textu. Mým úkolem bylo samotný algoritmus naimplementovat a provést na něm experimenty. Proto jsem pátral po implementacích určitých částí, o které se algoritmus opírá. Jedná se zejména o algoritmus α-expanze, potažmo grafových řezů, hledání maximálního toku apod.. Nalezl jsem a použil knihovnu, kterou autoři poskytují k článku [2].

Už jsem několikrát zmínil, že algoritmus potřebuje na vstupu, kromě samotného obrázku, ještě množinu záplat a transformací. Z toho důvodu jsem naimplementoval editor, který dovoluje uživateli záplaty vybrat a přidat jim potřebné transformace. Celou množinu záplat, včetně obrázku, pak umožňuje uložit do vlastního formátu a pak znovu načíst.

Výsledná aplikace má tedy dvě části. První je editor záplat a druhá je samotná segmentace, jejíž grafický výstup bylo možné vidět v předchozí kapitole.

Původně jsem jsem pro práci s obrázky při výpočtu algoritmu použil knihovnu OpenCV. Později jsem kvůli pohodlnějšímu testování začal tvořit grafický editor. Ten vyžadoval práci s GUI, a proto jsem pro něj použil knihovnu Qt. Aby byla implementace jednotná, přepsal jsem implementaci algoritmu za použití knihovny Qt.

Zdrojové kódy včetně programátorské dokumentace jsou dostupné na přiloženém CD v adresáři Code. Stručná uživatelská dokumentace je k dispozici přímo v aplikaci. Postup překladu zdrojových kódů a spuštění aplikace je v souboru Code/README.txt.

7 Závěr

Cílem diplomové práce bylo ověřit vlastnosti segmentační metody popsané v článku Gorelick L. et al.: "Recursive MDL via Graph Cuts: Application to Segmentation"(ICCV 2011). Mým úkolem bylo si článek a související články nastudovat, popsanou metodu naimplementovat v C/C++, provést na ni sérii experimentů a ty zdokumentovat v textové části práce. Podle okolností jsem se mohl pokusit metodu i vylepšit.

Úkoly jsem tedy začal podle pořadí plnit. Studium článků bylo poměrně náročné. V pozadí metody stojí několik dalších, nepříliš primitivních algoritmů. Některé jsem považoval pro hlavní metodu natolik podstatné, že jsem jim věnoval celou třetí kapitolu, kde jsem je poměrně podrobně popsal.

Hlavní metodě jsem věnoval čtvrtou kapitolu. V úvodu této kapitoly jsem se pokusil algoritmus shrnout pouze slovy a obrázky. Po něm však následoval nutný podrobný teoretický výklad, jehož čtení nemusí být úplně snadné.

Algoritmus se mi podařilo naimplementovat a výslednou aplikaci jsem rozšířil o grafický editor pro získání záplat. Několik slov k implementaci jsem shrnul v šesté kapitole. Aplikace by mohla být dále rozšířena o algoritmus pro automatický výběr záplat.

Teorie, na niž metoda stojí, mi přijde velice zajímavá. V praxi však velmi naráží na výpočetní výkon současných počítačů. Na vstup algoritmu je proto přivedeno jen několik záplat a při výpočtu je použit navíc hltavý přístup. Výsledek tedy není takový, jaký by mohl podle teorie být. I tak považuji výsledky, kterých jsem při testování dosáhl, za dobré.

Čas, který si výpočet vyžádal, na tom tak dobře není. Podívejme se na experiment s mrakodrapy (obrázek č. 5.9). V experimentu je testován obrázek pouze o rozměrech 300x217 pixelů. Dále je na vstupu 6 samostatných pixelů a pět záplat o rozměrech 30x30, 46x46, 62x62 a 75x75 pixelů. Doba potřebná pro výpočet (průměr deseti měření) na procesoru Intel C2D T5470 1.6 GHz byla 14280 ms. Výpočet by se dal urychlit např. paralelizací některých částí a implementací na grafické kartě. Nic to nemění na tom, že je výpočet časově velmi náročný.

Poslední částí bylo pokusit se algoritmus vylepšit. Napadlo mě vylepšení, že by algoritmus na vstupu, kromě záplat, které z něj pochází, mohl dostat i záplaty z jiných obrázků. To nijak neodporuje popsané teorii a myslím, že by to mohlo mít v určitých situacích přínos. Zejména, kdyby byl pro výběr záplat použit nějaký automatický algoritmus.

8 Reference

[1] GORELICK, L., A. DELONG, O. VEKSLER a Y. BOYKOV. Recursive MDL via Graph Cuts: Application to Segmentation. In: *Proceedings / IEEE International Conference on Computer Vision*. 2011, s. 890-897. ISSN 1550-5499.

[2] DELONG, A., A. OSOKIN, H.N. ISACK a Y. BOYKOV. Fast approximate energy minimization with label costs. In: *Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, 2173 - 2180. ISSN 1063-6919.

[3] BOYKOV, Y., O. VEKSLER a R. ZABIH. Fast approximate energy minimization via graph cuts. In: *IEEE transactions on pattern analysis and machine intelligence*. 1222 - 1239. ISSN 0162-8828.

[4] LOMBAERT, H. Energy minimization with Graph Cuts. *Herve Lombaert* [online]. [cit. 2012-03-06]. Dostupné z: http://step.polymtl.ca/ rv101/energy/

[5] BOYKOV, Y., O. VEKSLER a R. ZABIH. Efficient Approximate Energy Minimization via Graph Cuts. In: *IEEE transactions on pattern analysis and machine intelligence*. 2001, s. 1222-1239. 20: 12.

[6] KOLMOGOROV, V. a R. ZABIN. What energy functions can be minimized via graph cuts?. In: *IEEE transactions on pattern analysis and machine intelligence*. 2004, 147 - 159. 26: 2. ISSN 0162-8828.

[7] BOYKOV, Y. a V. KOLMOGOROV. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In: *IEEE transactions on pattern analysis and machine intelligence*. 2004, 1124 - 1137. 26: 9. ISSN 0162-8828.

[8] WANG, H., Y. WEXLER, E. OFEK a H. HOPPE. Factoring repeated content within and among images. In: *ACM Transactions on Graphics*. New York: Association for Computing Machinery, 2008, 1 - 10. 27: 3. ISSN 0730-0301.

[9] JOJIC, N., B. J. FREY a A KANNAN. Epitomic analysis of appearance and shape. In: *Ninth IEEE international conference on computer vision, 13-16 October 2003, Nice, France*. Los Alamitos (California): IEEE Computer Society, 2003, 34 - 44. 2. ISBN 0-7695-1950-4.

[10] BARNES, C., E. SCHECHTMAN, A. FINKELSTEIN a D. B. GOLDMAN. PatchMatch : A Randomized Correspondence Algorithm for Structural Image Editing. In: *ACM Transactions on Graphics*. New York: Association for Computing Machinery, 2009, 1 - 11. 28: 3. ISSN 0730-0301.

[11] GLASNER, D., S. BAGON a M. IRANI. Super-resolution from a single image. In: *IEEE 12th International Conference on Computer Vision*. 2009, 349 - 356. ISSN 1550-5499.

[12] Summarizing visual data using bidirectional similarity. In: SIMAKOV, D., Y. CASPI, E. SHECHTMAN a M. IRANI. *Ieee conference on computer vision and pattern recognition*. New York: Ieee Press Books, 2008, 1 - 8. ISBN 9781424422425.

[13] Towards Patient-Specific Heart Models for Decision Support System. *Health-e-Child deliverable D11.4* [online]. [cit. 2012-03-31]. Dostupné z: http://www-sop.inria.fr/asclepios/projects/Health-e-Child/DiseaseModels/content/cardiac/patientModels1_MRI.php

[14] Fingerprint recognition. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-01-12]. Dostupné z: http://en.wikipedia.org/wiki/Fingerprint_recognition

[15] Iris Recognition from 1-2 Meters. MITSUBISHI ELECTRIC RESEARCH LABORATORIES. *Mitsubishi Electric Research Laboratories* [online]. 2007-9-12 [cit. 2012-01-12]. Dostupné z: http://www.merl.com/areas/irisrecognition/

[16] Segmentation (image processing). In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-01-12]. Dostupné z: http://en.wikipedia.org/wiki/Segmentation_(image_processing)

[17] SOJKA, Eduard. *Digitální zpracování a analýza obrazů*. 1. vyd. Ostrava: VŠB - Technická univerzita, 2000, 133 s. ISBN 80-707-8746-5.

[18] BOYKOV, Y. a M.-P. JOLLY. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In: *Proceedings. Eighth IEEE International Conference on*. Vancouver, BC, 2001, 105 - 112. 1.

[19] LAWLER, Eugene L. *Combinatorial optimization*: *networks and matroids*. New York: Holt, Rinehart and Winston, c1976, s. 110-113. ISBN 0-03-084866-0.

[20] Max-flow min-cut theorem. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-02-01]. Dostupné z: http://en.wikipedia.org/wiki/Max-flow_min-cut_theorem

[21] VEKSLER, O. *Efficient graph-based energy minimization methods in computer vision*. Ithaca, NY, USA, 1999. ISBN 0-599-41334-4. PhD thesis. Cornell University.

[22] Texture segmentation. *The Department of Image Processing and Recognition* [online]. [cit. 2012-03-20]. Dostupné z: http://irtc.org.ua/image/pages/research/texture_segmentation

[23] Jan-Mark Geusebroek: research on computer vision and scene understanding. *University of Amsterdam, Faculty of Science* [online]. 2007 [cit. 2012-03-20]. Dostupné z: http://staff.science.uva.nl/ mark/downloads.html

[24] Project: Texture Segmentation using Gabor Filters and K-means Clustering. *Naotoshi Seo* [online]. 2006 [cit. 2012-04-29]. Dostupné z: http://note.sonots.com/SciSoftware/GaborTextureSegmentation.html

[25] Tulips of Holland. *The Best Travel Destinations* [online]. 2012 [cit. 2012-04-29]. Dostupné z: http://thebesttraveldestinations.com/tulips-of-holland/

[26] Thomas Struth: Photographs 1978-2010. *Harrison Cronbi* [online]. 2011 [cit. 2012-04-29]. Dostupné z: http://www.cronbi.com/2011/08/16/thomas-struth-photographs-1978-2010/stuth3/